VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

ÚSTAV SOUDNÍHO INŽENÝRSTVÍ
INSTITUTE OF FORENSIC ENGINEERING

# TAJMLAJN.CZ – WEBOVÁ SLUŽBA PRO PLÁNOVÁNÍ DOSAŽENÍ CÍLE A PŘEDCHÁZENÍ RIZIK

TAJMLAJN.CZ – WEB SERVICE FOR ACHIEVING THE OBJECTIVE OF PLANNING AND RISK PREVENTION

DIPLOMOVÁ PRÁCE
DIPLOMA THESIS

AUTOR PRÁCE
AUTHOR
BC. JAKUB LUDWIG

VEDOUCÍ PRÁCE
SUPERVISOR
ING. IGOR SZŐKE, PH.D.

BRNO 2014

Vysoké učení technické v Brně, Ústav soudního inženýrství

Ústav soudního inženýrství
Akademický rok: 2013/2014

# ZADÁNÍ DIPLOMOVÉ PRÁCE

student(ka): Bc. Jakub Ludwig

který/která studuje v **magisterském navazujícím studijním programu**

obor: **Řízení rizik v informačních systémech (3901T047)**

Ředitel ústavu Vám v souladu se zákonem č.111/1998 o vysokých školách a se Studijním a zkušebním řádem VUT v Brně určuje následující téma diplomové práce:

**Tajmlajn.cz - webová služba pro plánování dosažení cíle a předcházení rizik**

v anglickém jazyce:

**Tajmlajn.cz - Web Service for Achieving the Objective of Planning and Risk Prevention**

Stručná charakteristika problematiky úkolu:

Technická zpráva diplomové práce musí obsahovat formulaci cíle, charakteristiku současného stavu, teoretická a odborná východiska řešených problémů a specifikaci etap, které byly vyřešeny v rámci semestrálního projektu (30 až 40% celkového rozsahu technické zprávy).
Student odevzdá v jednom výtisku technickou zprávu a v elektronické podobě zdrojový text technické zprávy, úplnou programovou dokumentaci a zdrojové texty programů. Informace v elektronické podobě budou uloženy na standardním nepřepisovatelném paměťovém médiu (CD-R, DVD-R, apod.), které bude vloženo do písemné zprávy tak, aby nemohlo dojít k jeho ztrátě při běžné manipulaci.

Cíle diplomové práce:

1. Navrhněte webovou službu pro definování a sledování dosažených cílů. Jako příklad můžete uvažovat službu pro podporu řešení diplomových prací. Udělejte rešerši podobných služeb.
2. Implementujte základní verzi služby s ohledem na API a možnost navázání aplikací třetích stran - například mobilní aplikace.
3. Otestujte základní verzi služby s dostatečným množstvím skutečných uživatelů (například studentů řešících BP/DP).
4. Implementujte plnou verzi služby. Zaměřte se též na bezpečnostní aspekt služby a snažte se předejít možným hrozbám útoku a zneužití dat.
5. Otestujte chování uživatelů a reflektujte závěry z měření (změna GUI, vzhledu, funkcí, ...), diskutujte dosažené cíle a navrhněte směry dalšího vývoje.
6. Vyrobte A2 plakátek nebo cca 30 vteřinové video prezentující výsledky vaší práce.

Seznam odborné literatury:

Podle pokynů školitele

Vedoucí diplomové práce: Ing. Igor Szőke, Ph.D.

Termín odevzdání diplomové práce je stanoven časovým plánem akademického roku 2013/2014.

V Brně, dne 25.11.2013

L.S.

_____

doc. Ing. Aleš Vémola, Ph.D.
Ředitel vysokoškolského ústavu

***Abstrakt***

Tato diplomová práce popisuje vývoj aplikace Tajm-lajn, která slouží pro plánování času pro jednotlivce, nebo malé společnosti. Práce popisuje jednotlivé etapy vývoje a jejich testování uživateli a v závěru se snaží zhodnotit dosažený úspěch a budoucí vývoj celého projektu.

***Abstract***

This thesis describes the development of Tajm-lajn, application for effective and simple time planning in small business. The work contains detailed description of all development stages and user testing and in the conclusion it tries to summarize achieved results and future development of Tajm-lajn project.

***Klíčová slova***

Plánování času, čas, web, webové technologie, HTML, PHP, MySQL, JavaScript jQuery, uživatelské rozhraní, Lean development

***Keywords***

Time planning, time, web, web technology, HTML, PHP, MySQL, JavaScript, jQuery, user interface, Lean development

***Prohlášení***

Prohlašuji, že jsem diplomovou práci zpracoval/a samostatně a že jsem uvedl/a všechny použité informační zdroje.

V Brně dne ……..

…….

podpis diplomanta

***Poděkování***

Děkuji svému vedoucímu Ing. Igorovi Szőkemu, Ph.D. za inspiraci při tvorbě této práce a velké množství času, které mi věnoval na konzultacích.

# OBSAH

# ÚVOD

Diplomová práce, kterou právě držíte v rukou, si klade za cíl přiblížit čtenáři postupný vývoj aplikace Tajm-lajn (1), která má sloužit pro intuitivní plánování času na projektech. V dnešní době se vše velmi zrychluje a tak je efektivní plánování času stále důležitějším prvkem naší práce. Čtenář si asi může položit otázku, proč vymýšlet aplikaci pro plánování času, když jich dnes na trhu je velké množství. Mojí snahou není znovu vymyslet kolo, jen jsem nenašel aplikaci, která by splňovala mé požadavky. Dostupné aplikace pro plánování času fungují podobně jako aplikace pro operační systém Microsoft Windows. Snaží se dělat velkou spoustu věcí najednou a nabídnout tak uživateli cokoliv, na co jen pomyslí. Nápad to jistě není špatný, ale pokud chci pro sebe naplánovat čas a musím projít množstvím kroků, které naprosto nepotřebuji, tak mi aplikace ve výsledku čas spíše sebere, než aby učinila mou práci efektivnější. Pro naplánování času na příští týden si musím například vytvořit nový tým, projekt, specifikaci, milníky atd. Tohle ale já nepotřebuji. Proto jsem se rozhodl vytvořit aplikaci v souladu se zásadami operačního systému Linux. V Linuxu máte k dispozici velké množství aplikací a každá dělá přesně jednu věc. Nesnaží se jich dělat velké množství najednou, jen tu jednu, ale pořádně a kvalitně. Chcete například zjistit kolik je v souboru řádků s konkrétním jménem? Jeden příkaz vám najde řádky, druhý je spočítá.

V souladu s touto filozofií se budu snažit navrhnout mojí aplikaci pro plánování času. Chci od ní, aby uměla pouze plánovat čas na určitém projektu, nepotřebuji úložiště souborů, management zdrojů, finance atd. Inspirací pro mou diplomovou práci je wiki stránka s časovou osou, kterou používá vedoucí mé práce Ing. Igor Szőke, Ph.D. právě pro plánování a monitorování času, který jeho diplomanti stráví na jejich diplomových pracích.

Zároveň bych velmi rád použil mou diplomovou práci i v komerčním světě a zkusil kolem ní vybudovat start-up, takže při návrhu aplikace musím přemýšlet i nad komerční částí. Zpětná vazba uživatelů proto pro mě bude velmi důležitá.

# 1 POUŽITÉ TECHNOLOGIE

V této kapitole bych se rád věnoval technologiím, které použiji pro realizace mé diplomové práce. Jelikož zadání mé práce je vytvořit webovou aplikaci, zvolil jsem klasickou kombinaci technologií, která je dnes velmi běžná, snadno dostupná, zdarma a otestovaná velkou komunitou uživatelů z celého světa.

## 1.1 APACHE

Jako webový server jsem zvolil Apache (2), který vyvíjí společnost The Apache Software Foundation. Tento webový server je zdarma a dnes ho najdeme na velkém množství serverů, které zprostředkovávají webové stránky. Díky jeho rozšířenosti je pro něj dostupné velké množství různých modulů a rozšíření. Pro účely mé práce budu potřebovat dva z těchto modulů a to modul pro interpretaci jazyka PHP a modul pro práci s databází MySQL.

Další nespornou výhodou webového serveru Apache je fakt, že je multiplatformní a tak není problém ho spustit na operačním systému Linux, Microsoft Windows 7 nebo FreeBSD.

## 1.2 HTML A CSS

Jelikož výstupem mé diplomové práce má být webová aplikace, neměl jsem při výběru technologie pro zobrazení stránek moc na výběr. Značkovací jazyk HTML[3] je standardem pro webové stránky. Jazyk HTML dodnes vydává a směr jeho vývoje určuje W3C konsorcium (3).

Samotné HTML ale dnes už pro zobrazování stránek nestačí a musím použít ještě technologii CSS[4], která umožňuje definovat vzhled jednotlivých elementů webové stránky v odděleném souboru a prakticky tak rozděluje formu od obsahu, kde v jednom souboru je čisté HTML a v druhém jsou uloženy styly, které definují, jak bude který prvek vypadat.

## 1.3 PHP

Jako programovací jazyk jsem zvolil jazyk PHP (4). Zkratka PHP znamená PHP: Hypertext Preprocessor a je rekurzivní (žertík autorů, první písmeno zkratky je zároveň zkratkou samotnou). Jazyk PHP je dnes jedním z nejrozšířenějších jazyků pro tvorbu webových

[3] HyperText Mark-up Language

[4] Cascading Style Sheet

stránek a širokou podporu v mnoha webových serverech. Mezi hlavní rysy jazyka PHP patří slabě typované proměnné, objektové orientovaný přístup, použití výjimek, jmenných prostorů a rozhraní. Obzvlášť díky slabě typovaným proměnným je tento jazyk velmi rozšířen mezi široké spektrum programátorů, protože nevyžaduje tak striktní pravidla při programování a uživatel si nemusí moc lámat hlavu s typem proměnné a alokovanou pamětí.

Velkou výhodou PHP je velmi aktivní komunita, díky které je na internetu obrovské množství návodů, popisů, dokumentace a hotových knihoven. Typické problémy, které se uživatel snaží řešit, mají zdokumentované to nejlepší řešení přímo na stránkách PHP (anglicky best practices).

## 1.4 MYSQL

Pro ukládání dat v mé diplomové práci použiji databázi MySQL (5), která je volně dostupná a dobře se doplňuje s jazykem PHP, který pro ni má nativní podporu. Databáze MySQL poskytuje všechny důležité funkce, jako integritní omezení, cizí klíče a triggery. MySQL nabízí několik formátů úložiště dat, které mohou být použity pro tvorbu tabulek. V mojí práci budu používat formát InnoDB (6), který podporuj již zmiňované použití cizích klíčů a referenční integritu.

## 1.5 JAVASCRIPT

JavaScript (původně pojmenovaný LiveScript a později přejmenovaný kvůli marketingu) je skriptovací jazyk s prototypovou dědičností, který se dnes používá především pro zvyšování použitelnosti a uživatelské přívětivosti webových stránek. Interpretrem JavaScriptu je prohlížeč uživatele, takže to dává vývojářům možnost přesunout část zátěže ze serveru na klientský počítač. Dříve byl JavaScript používán k různým vtipným efektům (například oči létající za kurzorem, sníh na webové stránce atd.), nebo k zobrazení různých potvrzovacích dialogů, ale jak se postupně zvyšoval výkon uživatelských počítačů, bylo možné pomocí JavaScriptu vytvářet stále komplikovanější aplikace. Velký rozmach získal JavaScript díky technologii AJAX[5].

AJAX umožňuje vykonat asynchronní dotaz na webový server a následně zpracovat data, která server odešle. Výsledkem je tedy možnost navigovat se na webové stránce bez

[5] Asynchronous JavaScript and XML

nutnosti ji obnovovat a značně se snižuje zátěž na server, jelikož tento nemusí s každým požadavkem posílat vždy celou webovou stránku, ale jen tu část, kterou si JavaScript vyžádá. Dobrým příkladem opravdu masivního využití technologie AJAX je sociální síť Facebook (7).

Bohužel JavaScript má i své nevýhody, asi hlavní je rozdílná interpretace některých příkazů v různých internetových prohlížečích. Tento neduh obvykle znamená výrazně větší množství testování a optimalizací kódu, protože snahou vývojáře je přiblížit jeho produkt co nejširšímu spektru uživatelů. V případě JavaScriptu se občas můžeme setkat s pojmem „graceful degradation“. Tento pojem označuje fakt, že stránka funguje i s vypnutým JavaScriptem (klient si ho může v prohlížeči vypnout), jen například nevypadá tak hezky, nebo některé funkce nejsou tak pohodlné.

## 1.6 KNIHOVNA JQUERY

V případě JavaScriptu jsem se pro implementaci mé diplomové práce rozhodl použít knihovnu jQuery (8) od sdružení The jQuery Foundation (9). Úmyslně ji uvádím mezi použité technologie, protože tato knihovna do velké míry mění způsob s jakým se s JavaScriptem pracuje. Vytváří obal na většinu JavaScriptových funkcí a zajišťuje, že budou v každém prohlížeči vykonány stejně. Zároveň velmi usnadňuje práci například s technologií AJAX, pro porovnání, v čistém JavaScriptu potřebuje programátor cca 50 řádků kódu pro vykonání dotazu na server, díky jQuery stačí cca 3.

Knihovna jQuery je také známá pro časté využití lambda funkcí, které dělají, při správném použití, kód mnohem přehlednější a čitelnější. Existuje pro ni také velké množství rozšíření a návodů.

# 2 ANALÝZA EXISTUJÍCÍCH PHP FRAMEWORKŮ

V této kapitole bych se rád věnoval analýze existujících PHP frameworků, jejich představení, srovnání a nakonec výběru frameworku, ve kterém budu práci realizovat. A co to vlastně ten framework je? Pod pojmem framework si můžeme představit ucelenou sadu nástrojů, knihoven, pomocných skriptů a hlavně myšlenku, se kterou je autor (nebo autoři) dali dohromady právě tímto způsobem. Velmi často se myšlenka frameworku opírá o nějaký návrhový vzor (obvykle ne jen o jeden) a přichází se sadou konvencí, které by měl uživatel dodržovat, pokud chce framework používat. Odměnou je mu potom výrazné ušetření práce v mnoha oblastech vývoje, kde by jinak musel vše implementovat sám a od začátku, čímž může zanést do programu velké množství chyb. Míra závislosti na těchto konvencích je individuální pro každý framework.

Frameworky můžeme nalézt v téměř každém programovacím jazyce, který umožňuje dělit kód do tříd nebo do modulů. A nemusíme se nutně omezit jen na programovací jazyky, například pro jazyk CSS existuje také framework, který uživateli usnadňuje stylování webových stránek

## 2.1 NÁVRHOVÝ VZOR MODEL-VIEW-CONTROLLER

Jelikož většina moderních PHP frameworků využívá návrhového vzoru MVC[6], bylo by dobré vysvětlit, co je jeho podstatou. MVC dělí aplikaci na tři logické vrstvy, jak je vidět na obrázku 3.1.

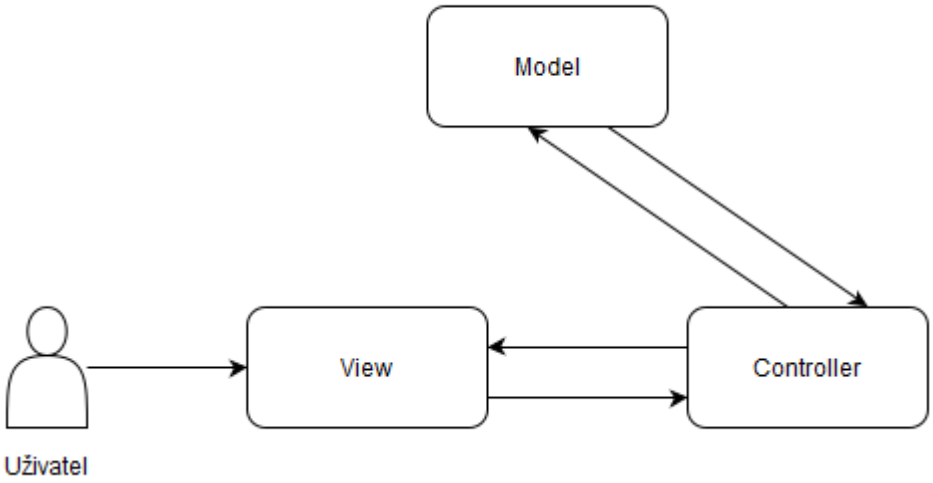
Uživatel zadá požadavek na vrstvu View. Vrstva View je to, co uživatel vidí, v případě webové aplikace je to odpověď z webového serveru, obvykle HTML stránka, kterou zobrazuje uživatelům prohlížeč. Zadání požadavku je například kliknutí na odkaz, odeslání formuláře, nahrání souboru atd. Vrstva View tento požadavek předá Controlleru, což je část aplikace, který obsahuje její logiku, co se má stát pro který daný požadavek. Controller může, nebo nemusí, kontaktovat Model. Model je vrstva, která se stará o data a neví nic o tom, jak jsou data ve

[6] Model – View – Controller

výsledku reprezentována, jestli jako HTML, nebo jako cokoliv jiného. Controller zpracuje uživatelův požadavek a předá patřičná data vrstvě View, která výsledek zobrazí.

V praxi to pak vypadá tak, že ve vrstvě Model najdeme obvykle dotazy do databáze, případně další práci s daty, v Controlleru je logika aplikace v tom smyslu, že ví, který model je třeba zavolat a jaká data získat. Vrstva View je obvykle systém šablon, ve kterém je pomocí speciálního jazyka definováno místo (nebo místa), kam se budou data z Controlleru vkládat a jakým způsobem.

Velkou výhodou tohoto rozdělení je fakt, že například můžete nechat tvorbu šablon na někom, kdo skvěle ovládá HTML, ale neví vůbec nic o programování v PHP nebo databázích. Zároveň tento přístup dělá aplikace programátorsky přehlednější, protože nutí vývojáře rozdělit kód do logických celků.




*Obrázek 2.1 Diagram návrhového vzoru MVC ve webové aplikaci*

## 2.2 ZEND FRAMEWORK

Zend framework (10) patří mezi nejstarší PHP frameworky vůbec, dodnes je velmi aktivně vyvíjen firmou Zend Technologies (11) a jeho vývoj jde stále kupředu. Základní vlastností Zend frameworku je jeho velmi těsná provázanost s jazykem PHP, protože zakladatelé Zend Technologies jsou zároveň jedni z hlavních vývojářů jazyka PHP. To zaručuje velkou míru porozumění a také možnost značné ovlivňovat vývoj jazyka PHP jako takového.

Zend framework je kombinací jádra, které je nazváno Zend Engine a přidaných knihoven. První verze jádra pochází z roku 1999, ale celý framework dostal ucelenou podobu až v roce 2006, nicméně jednalo se pouze o marketingový tah, kvůli sjednocení všech produktů. Pro Zend framework je totiž k dispozici také vlastní webový server Zend Server a vlastní vývojový nástroj Zend Studio a navíc ještě aplikace Zend Guard pro ochranu zdrojového kódu před nelegálním šířením. Samotný framework je distribuován zdarma jako pro osobní, tak pro komerční využití, ale další nástroje už jsou samozřejmě placené. Proto vidím využití Zend frameworku spíše v korporátním prostředí, kde se očekává běh na výkonných serverech, které zvládnou přístup mnoha uživatelů najednou.

Celý Zend je postaven kolem návrhového vzoru MVC a v některých směrech velmi striktně vyžaduje jeho využívání. Jeho další velkou výhodou je, díky době jeho působení na trhu, velké množství hotových aplikací, které se dají buď rovnou použít, nebo se z nich dá dobře učit. Na internetu je možno nalézt velké množství tutoriálů pro práci se Zend frameworkem a jeho knihovnami, což považuji za velké plus. Jako nevýhodu vidím jeho vyšší nároky na webový server a jeho nastavení (vývojáři totiž preferují, aby Zend běžel na jejich vlastním Zend Serveru), takže u některých webhosting společností může být se zprovozněním Zendu problém.

## 2.3 CODEIGNITER

Framework CodeIgniter (12), za kterým stojí společnost EllisLab (13), o sobě tvrdí, že je to framework s nejmenším dopadem na už existující kód a jeho integrace do stávajícího kódu je nesmírně jednoduchá. Protože to není úplně moje kritérium výběru, tak tuto vlastnost nebudu brát v úvahu (tvořím aplikaci od začátku, nepotřebuji integrovat framework do existujícího kódu) a porovnám pouze funkční vlastnosti frameworku.

Celý framework je, na rozdíl od Zend framework pojat velmi liberálně a místo frameworku tak připomíná spíše sadu užitečných funkcí a knihoven, které uživatel může použít.

Stejně jako Zend, i CodeIgniter podporuje návrhový vzor MVC, ale má velmi volná pravidla pro jeho implementaci, takže například dovoluje uživateli naprosto vynechat vrstvu Model a místo ní všechnu práci s daty provozovat ve vrstvě Controller.

Také zprovoznění celého frameworku je otázka několika minut, stačí stáhnout zabalený soubor, rozbalit, nastavit cestu k aplikaci a k databázi a vše je připraveno k použití. Na rozdíl od Zend frameworku, kde celé nastavení aplikace je poněkud komplikovanější, je CodeIgniter více cílem na menší společnosti a jednotlivé programátory. Zároveň ale nenabízí žádné další nástroje ani vlastní webový server, jako je tomu u Zend frameworku. Další nevýhodou je poměrně malé množství hotových aplikací a ukázek kódu. Oficiální dokumentace je kvalitní, ale mimo ní je problém sehnat další informace.

## 2.4 NETTE

Poslední vybraným zástupcem je Nette framework (14) od českého autora Davida Grudla a sdružení Nette Foundation (15). Framework vznikl v roce 2005 a od té doby se na jeho vývoji podílí jak jeho zakladatel, tak celá komunita, která za tu dobu kolem frameworku vznikla. Nette podporuje (stejně jako Zend a CodeIgniter) návrhový vzor MVC, ačkoliv je, stejně jako CodeIgniter, značně benevolentní k použití vrstvy Model. Pro vrstvu View obsahuje vlastní šablonovací systém nazvaný Latte, který je s celým frameworkem integrován a nabízí tak kromě základních funkcí, jako je nahrazování obsahu z Controlleru, také další pokročilé funkce, jako je například předgenerování Latte souborů do čistého PHP, což významně urychluje vykreslování stránky.

Mezi další velké výhody Nette patří fakt, že framework od začátku počítá s podporou technologie AJAX, takže vytvoření aplikace, která může bez problémů fungovat bez načtení stránky, je zde otázka několika minut. Stejně jako ostatní frameworky obsahuje velké množství různých knihoven a rozšíření pro práci s formuláři, e-maily, vyrovnávací pamětí, soubory, obrázky a mnohým dalším.

Samotnou kapitolou jsou pak ladící nástroje. S Nette je dodávána komponenta DebuggBar, kterou ovšem autor frameworku familiárně překřtil na „*Laděnku*“. Tato komponenta se zobrazuje v pravém dolním rohu stránky jako malý panel, který informuje uživatele o době potřebné pro vykreslení stránky, přihlášeném uživateli, množství výkonaných dotazů do databáze a mnoho dalšího. DebuggBar je navíc rozšiřitelný o vlastní komponenty díky definovanému programovému rozhraní pro psaní těchto komponent. Hlavní výhoda této komponenty se ale projeví v okamžiku, kdy dojde k chybě v PHP. Místo klasické jednořádkové

chyby, jejíž vypovídající hodnota je obvykle velmi malá, se objeví obrazovka, která zobrazuje soubor, ve kterém došlo k chybě, barevně označí řádek a dá programátorovi mnoho užitečných informací o běhu programu. Velmi jednoduchým způsobem je programátorovi vizualizován postup volání jednotlivých procesů tzv. stacktrace, který je také interaktivní. Dále je k dispozici výpis všech proměnných, definovaných v době pádu programu, výpis hyperglobálních[7] proměnných $_SERVER, $_POST, $_GET a dalších.

Mezi nevýhody Nette řadím jeho dokumentaci, která je mnohdy velmi zmatená, nebo nedostačující. K dispozici je sice kompletní vygenerovaná dokumentace pro jednotlivé třídy a metody, ale z hlediska uživatelské dokumentace má ještě framework co dohánět. Další nevýhodou může být zpočátku zmatené pojmenování některých částí frameworku (například Controller se zde jmenuje Presenter, protože to autorovi přišlo výstižnější).

## 2.5 ZHODNOCENÍ A VÝBĚR FRAMEWORKU

Po pečlivém zvážení všech pro a proti jsem se nakonec rozhodl zvolit Nette framework a to především pro jeho eleganci a nutnost psát „*hezký*“ kód, protože Nette vás nutí dodržovat určité postupy, díky kterým je kód lépe strukturovaný a dává větší smysl. Obrovskou výhodou jsou také ladící nástroje, kterými je Nette vybaveno, a které pro mě znamenají velké plus. Dalším důvodem jsou mé zkušenosti s prací v Nette (aktuálně 2 roky), takže věřím, že aplikace bude napsaná výrazně lépe, než kdybych se měl učit nějaký jiný framework.

[7] Proměnné, které jsou dostupné v každém skriptu globálně

# 3 ANALÝZA EXISTUJÍCÍCH APLIKACÍ PRO PLÁNOVÁNÍ ČASU

Než jsem započal práci na vlastní aplikaci, bylo zapotřebí provést analýzu dostupných řešení. Na internetu se jich dá najít opravdu velké množství, nicméně chtěl jsem nějaké zástupce, které uživatelé opravdu používají. Proto jsem vytvořil dotazník na téma „*Jaké znáte služby pro plánování času*“ a rozšířil ho mezi mé přátele a kolegy v jiných firmách. Výsledkem bylo 51 odpovědí, ze kterých jsem si vybral tři nejčastější aplikace a tyto jsem vyzkoušel a podrobněji analyzoval.

## 3.1 BASECAMP

Jeden z nejznámějších nástrojů od firmy 37signals je k nalezení na adrese http://basecamp.com a snaží se uživatelům nabídnout kompletní management projektů od událostí v kalendáři, přes diskuze, možnost přidávání souborů, více projektů, týmů a další. Projekt má dvou měsíční trial verzi, která je zdarma, následně je zpoplatněn podle počtu projektů a zabraného místa. Po přihlášení je v aplikaci dashboard, který informuje o všem důležitém v projektu. Basecamp také umožňuje přizvat do projektu přímo zákazníka a dává možnost ukázat mu některé části systému, například naplánované schůzky, úkoly atd.

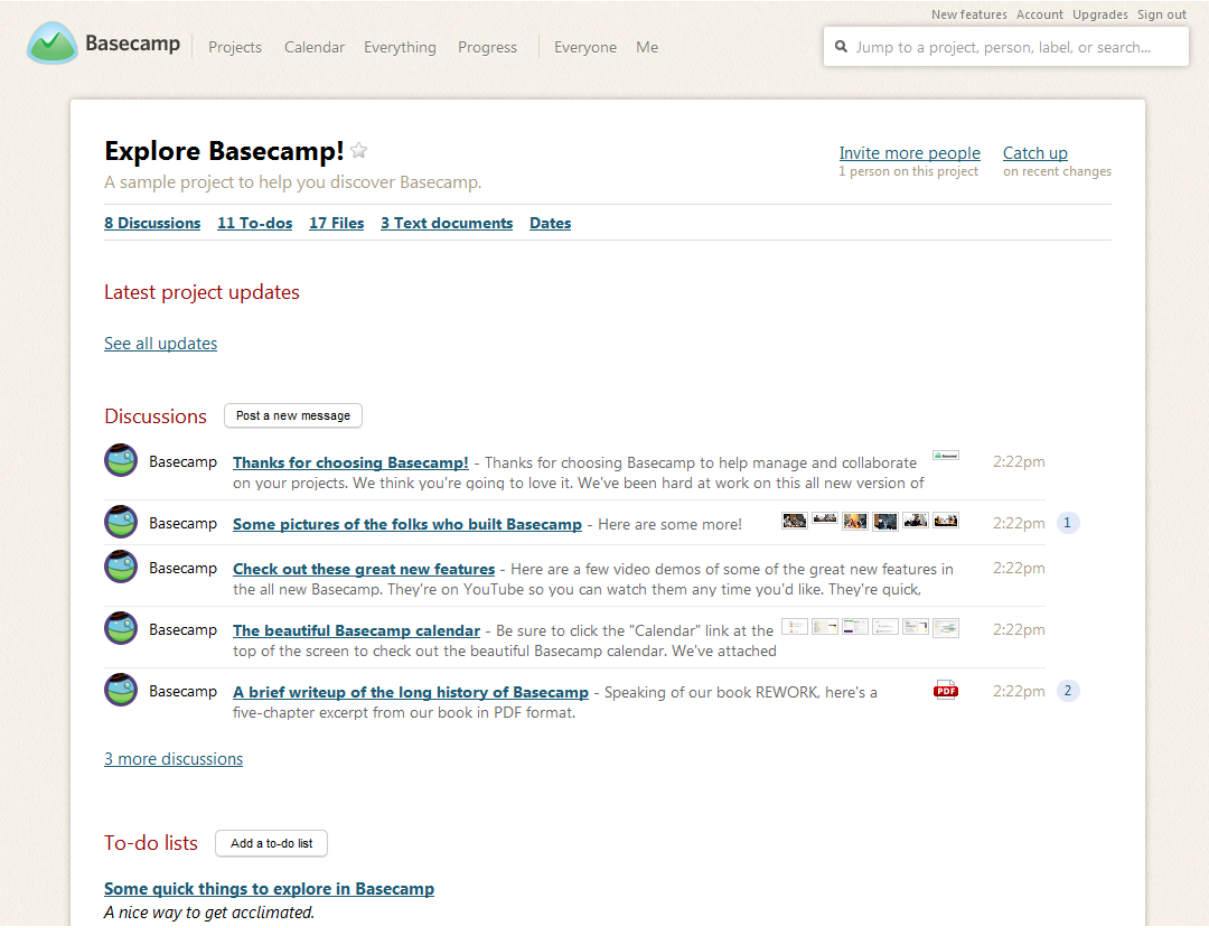
Pro každý nový projekt je možnost vytvořit jakousi šablonu, podle které se mohou tvořit další projekty. Pokud tedy mám nějaká nastavení typická pro každý projekt, je dobré si vytvořit šablonu, značně to šetří čas. Otázkou zůstává, jak často potřebuje uživatel vytvářet nové projekty, aby mělo smysl dělat šablonu.

### 3.1.1 Výhody a nevýhody

Mezi hlavní výhody Basecampu bych zařadil částečnou integraci s dalšími projekty společnosti 37signals, které slouží pro správu kontaktů a tvorbu on-line diskuzní místností. Do jisté míry to odpovídá mé myšlence dělat jednu věc, ale pořádně, než více věcí najednou. Další výhodou je přizpůsobení aplikace pro mobilní telefony a tablety, ať už formou responsivního designu, nebo nativní aplikace přímo pro mobilní telefon, nebo tablet (Apple iOS).

Mezi nevýhody podle mého názoru patří přílišná komplikovanost uživatelského rozhraní. Na první pohled po přihlášení sice vidím projekty, ale po kliknutí na některý projekt dostanu již zmiňovaný dashboard, který je ale velmi nepřehledný a preferoval bych možnost si

ho upravit podle svého přání. Například možnost oddělit věci barvou, nebo je uspořádat podle svého přání mi velmi chybí.




*Obrázek 3.1 Ukázka uživatelského rozhraní aplikace Basecamp*

## 3.2 TEAMBOX

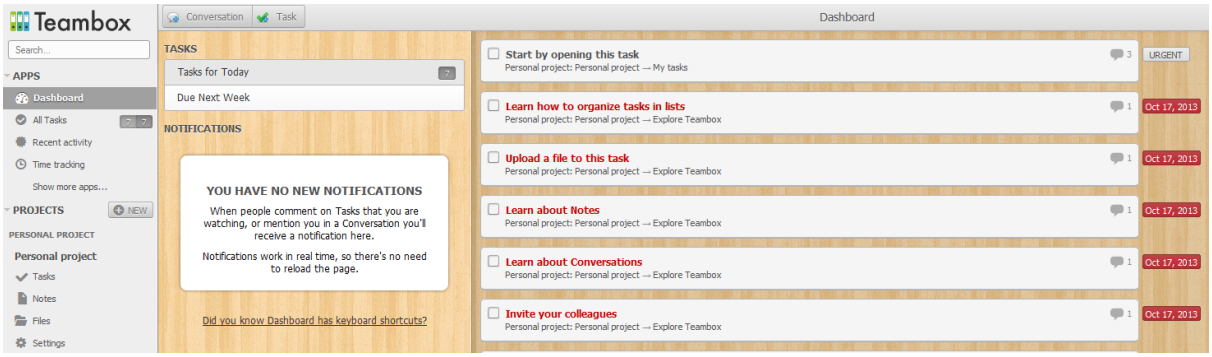
Aplikace Teambox má být podle informací na internetových stránkách https://teambox.com celá organizovaná pomocí úkolů, kolem kterých se má v aplikaci točit vše ostatní. Rozhodl jsem se ji vyzkoušet, protože nabízí trial verzi zdarma, kde některé rozšířené funkce nejsou dostupné. Rozhraní se snaží tvářit příjemně, ale ve skutečnosti je značně nepřehledné. Tam, kde měl Basecamp velká jednoduchá tlačítka, tam má Teambox klikatelný text, nebo malé šedivé tlačítko. Na webu uvádějí, že je možné diskutovat s kolegy ve stejném týmu buď pomocí textových diskuzí, nebo High Definition video konference. Tuto možnost jsem bohužel neměl možnost vyzkoušet, ale zdá se mi zajímavá.

Ke každému úkolu můžete někoho přiřadit, určit do kdy má být hotový, jakou má prioritu a připojit k němu soubor. Soubor může být buď nahrán z počítače, nebo vložen z některé online služby pro ukládání souborů, ať už se jedná o Google Disk nebo populární službu Dropbox.

### 3.2.1 Výhody a nevýhody

Jako velkou výhodu vidím integraci s Google Disk a Dropbox pro sdílení souborů, značně to odstraňuje zbytečnou redundanci souborů, kdy pak nevím, jestli je aktuální soubor na Dropboxu, nebo nahraný v Teamboxu. Zároveň tak můžu mít všechny souboru na jednom místě a mít jen jednu aplikaci, která se mi stará o sdílení těchto souborů se zbytkem týmu. Další výhodou je možnost zadávat a spravovat úkoly pomocí e-mailového klienta, pomocí speciálních odpovědí na určený e-mail na Teamboxu.

Jako hlavní nevýhodu vidím celkovou nepřehlednost uživatelského rozhraní a nutnost udělat mnoho zbytečných kroků pro něco tak jednoduchého, jako vytvoření úkolu (který má být základem celé aplikace).

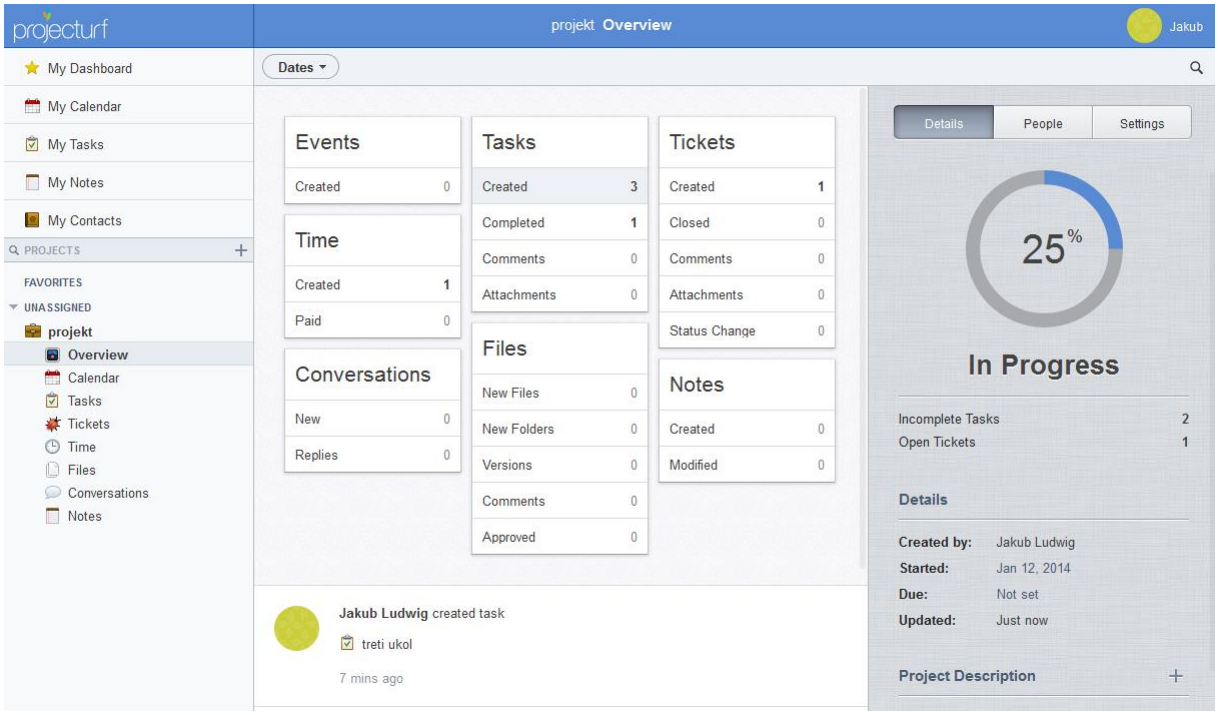



*Obrázek 3.2 Ukázka uživatelského rozhraní aplikace Teambox*

## 3.3 PROJECTURF

Aplikaci Projecturf nalezneme na adrese http://projecturf.com/ a nabízí téměř to samé, co ostatní dvě aplikace, tedy možnost zakládat projekty, do projektů připojovat úkoly, soubory, konverzace, sestavit tým kolegů a dokonce přidávat odpracovaný čas (byť velmi jednoduchým způsobem). Celá aplikace má velmi jednoduché a přehledné uživatelské rozhraní, které sice místy působí stroze, ale účel splňuje.

Na rozdíl od předchozích dvou konkurentů zde nenajdeme žádnou možnost propojení s jinou službou, jako například Dropbox pro sdílení souborů, ani žádnou mobilní aplikaci. Z tohoto pohledu Projecturf značně zaostává za konkurencí. Ze všech třech testovaných produktů je také nejdražší.




*Obrázek 3.3 Ukázka uživatelského rozhraní aplikace Projecturf*

### 3.3.1 Výhody a nevýhody

Mezi velké výhody řadím na prvním pohled přehledné uživatelské rozhraní (jak je vidět na obrázku 4.3), kde je jasně vidět kolik mám rozdělaných úkolů, poznámek, událostí, co se stalo naposled a jako příjemný bonus je sumář po pravé straně aplikace.

Bohužel uživatelské rozhraní je přívětivé jen v některých ohledech. Rozhodně je přehledné, ale například na označení projektového ticketu jsem si musel vzít k ruce manuál,

abych našel možnost jak ho přesunout ze stavu Nové do stavu Hotovo. Jako další nevýhodu vidím nulovou podporu pro mobilní zařízení. Na diskuzních fórech projektu sice vývojáři tvrdí, že její vydání už se blíží, ale úspěšný produkt tohoto ražení si, dle mého názoru, nemůže dovolit ignorovat stále větší zastoupení mobilních platforem u koncových zákazníků.

## 3.4 KLASICKÝ KALENDÁŘ

Ve srovnání existujících řešení pro plánování času samozřejmě nesmí chybět ten nejstarší nástroj a tím je obyčejný kalendář. Kalendář dnes používá každý od studentů po top manažery a je to tedy naprosto validní řešení pro plánování času, ať už se jedná o kalendář elektronický nebo papírový.

Má aplikace by měla řešit i motivaci uživatele plnit naplánované úkoly a pohled do kalendáře plného úkolů podle mého názoru příliš motivační není. Zobrazené budou pouze úkoly v nejbližším časovém horizontu. Díky tomuto zobrazení je aplikace cílena na jednotlivce, nebo malé skupinky, které nepotřebují mít širší rozhled budoucích událostech. V případě nutnosti plánovat s velkým časovým předstihem je efektivnější použít klasický kalendář.

### 3.4.1 Výhody a nevýhody

Hlavní výhodou kalendáře je bezesporu jeho rozšířenost. Kalendář zná dnes téměř každý, kdo si někdy potřeboval plánovat čas a z hlediska přehlednosti mu jen máloco může konkurovat.

Nevýhodou je složitější sdílení kalendáře s kolegy a případná kolaborace a dále poněkud demotivační prvek, kdy člověk ráno otevře kalendář a vidí, co všechno ho ten den čeká a má tendence věci odkládat. Pro autora této práce je právě každodenní pohled do plného kalendáře největší motivací přijít s jiným řešením.

## 3.5 REKAPITULACE

Po pečlivém otestování zmiňovaných produktů jsem se rozhodl poučit se z jejich vývoje a vzít si z nich to, co se mi líbí a přijde mi užitečné a integrovat to do mé aplikace. Na všech třech aplikacích je vidět, že snaha dělat více věcí najednou je sice někdy užitečná, ale daň za to je často značně zmatené uživatelské rozhraní, ve kterém se uživatel ztrácí a které nepomáhá.

Ve své aplikaci proto budu klást velký důraz na návrh uživatelského rozhraní a budu ho při vývoji neustále konzultovat s potenciálními uživateli, abych se výsledným produktem co nejvíc přiblížil jejich požadavkům.

Dalším poznatkem je, že si nemohu dovolit ignorovat uživatele chytrých telefonů a tabletů. Aplikace proto musí mít minimálně responsivní design (design, který se přizpůsobuje velikosti displeje zařízení), ideálně pak vlastní webovou aplikaci, minimálně pro mobilní OS Android.

Pouze jedna ze tří testovaných aplikací umožňovala přihlášení jiným než vlastním účtem (například pomocí Google). Já osobně preferuji možnost používat jeden účet pro více aplikací, nemusím si pak pamatovat pro každou službu jméno účtu a heslo. Od mé aplikace proto budu požadovat, aby umožňovala použití existujícího účtu u některé z velkých internetových společností.

# 4 ANALÝZA A SPECIFIKACE POŽADAVKŮ

Po výběru technologií a analýze existujících řešení přichází na řadu specifikace požadavků na mojí vlastní aplikaci. Při specifikaci těchto požadavků vycházím z kapitoly 3 a z potřeb potenciálních uživatelů mé aplikace. Abych tyto potřeby zmapoval, rozhodl jsem se vytvořit dotazník a rozšířit ho mezi potenciální uživatelé mé aplikace. Na základě tohoto dotazníku jsem navrhl první etapu vývoje.

Pro vývoj jsem se rozhodl použít techniku LSD[8], která se řadí mezi agilní metodologie vývoje softwaru, a její základní principy můžete shrnout do několika bodů:

1. Eliminuj zbytečný kód
2. Rozhoduj se co nejpozději
3. Vydej produkt co nejrychleji, ať získáš zpětnou vazbu
4. Vnímej produkt jako celek

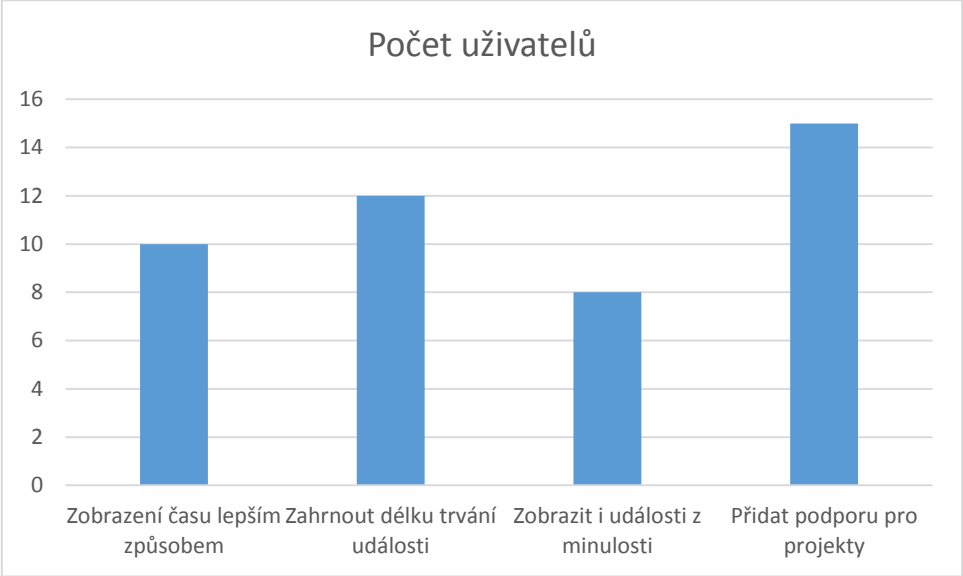
LSD také klade velký důraz na zpětnou vazbu zákazníka, kterou bych měl reflektovat v dalších iteracích vývoje. Čím rychleji je iterace dokončena, tím rychleji získá vývojář zpětnou vazbu a může produkt upravit podle očekávání zákazníků.

## 4.1 DOTAZNÍK

Dotazník jsem se rozhodl rozdělit na dvě části, komerční a technickou. Vzhledem k charakteru této práce zde nebudu uvádět výsledky komerční části a zaměřím se čistě na část technickou, tedy to, co uživatel od aplikace očekává. Celkem jsem získal odpovědi od 57 respondentů, mezi kterými jsou jak moji spolužáci, tak moji kolegové ze zaměstnání a další lidé, u kterých jsem doufal, že moje aplikace bude mít smysl (například top manažer jedné technologické firmy se sídlem v Brně).

Vzhledem k analýze existujících aplikací v kapitole 3 jsem se rozhodl postavit dotazník v takovém duchu, aby zjistil, jestli to, co jsem díky analýze zjistil a chtěl bych implementovat, má opravdu smysl. První otázkou proto bylo, jestli by uživatelé uvítali i mobilní verzi aplikace a pokud ano, tak pro jakou platformu. Z obrázku 5.1 je jasně patrné, že zájem ze strany uživatelů rozhodně je, 41 respondentů z 57 odpovědělo, že by uvítali dostupnost pro mobilní platformu Android.

[8] Lean Software Development




*Obrázek 4.1 Preference mobilních platforem*

Další otázkou bylo, zda by uživatelé uvítali možnost použít už existující účet u jiné společnosti pro přihlášení do mé aplikace. Z mého výzkumu je také patrné, že bych měl implementovat minimálně metodu přihlášení přes účet u společnosti Google.

Následovaly otázky na konkrétní technologie, které nemají na specifikaci požadavků žádný vliv, proto je neuvádím. V dotazníku byla ještě jedna část, kde jsem dal uživatelům možnost vlastních textových odpovědí a požádal jsem je, aby napsali pár věcí, které by, podle jejich názoru, v aplikaci rozhodně neměly chybět a které by aplikace naopak rozhodně neměla obsahovat.

Jako nejčastější požadavky na to, co v aplikaci rozhodně nesmí chybět, jsem vyhodnotil následující body:

1. Přehledné a jednoduché uživatelské rozhraní
2. Možnost sdílet své naplánované úkoly s dalším uživatelem
3. Notifikace na úkoly a události, ať už e-mailem, nebo uvnitř aplikace
4. Zapojení gamifikačních principů
5. Propojení s nějako službou pro sdílení souborů (podobně jako produkt Basecamp v kapitole 3.1)

Body jsou řazeny podle četnosti, s jakou se objevovaly v odpovědích respondentů, a jedná se o velmi podobné požadavky, jaké jsem měl i já po analýze existujících řešení.

Ten samý žebříček jsem sestavil i pro to, co by se podle mých respondentů v aplikaci rozhodně objevit nemělo:

1. Zbytečně složité ovládání
2. Reklamy
3. Vysoké požadavky na výkon uživatelova zařízení

Tento seznam velmi dobře doplňuje seznam toho, co by v aplikaci rozhodně nemělo chybět a stává se jeho komplementem v tom smyslu, že například bod 1 v obou seznamech je vlastně identický požadavek, jen vždy podaný jinak. Bodem 2 mi dávají uživatelé jasně najevo, že pokud má mít jednou služba komerční využití, tak rozhodně ne pomocí reklam, ale spíše pomocí předplatného. Požadavek na nízké nároky na zařízení je zaměřen spíše na mobilní verzi aplikace.

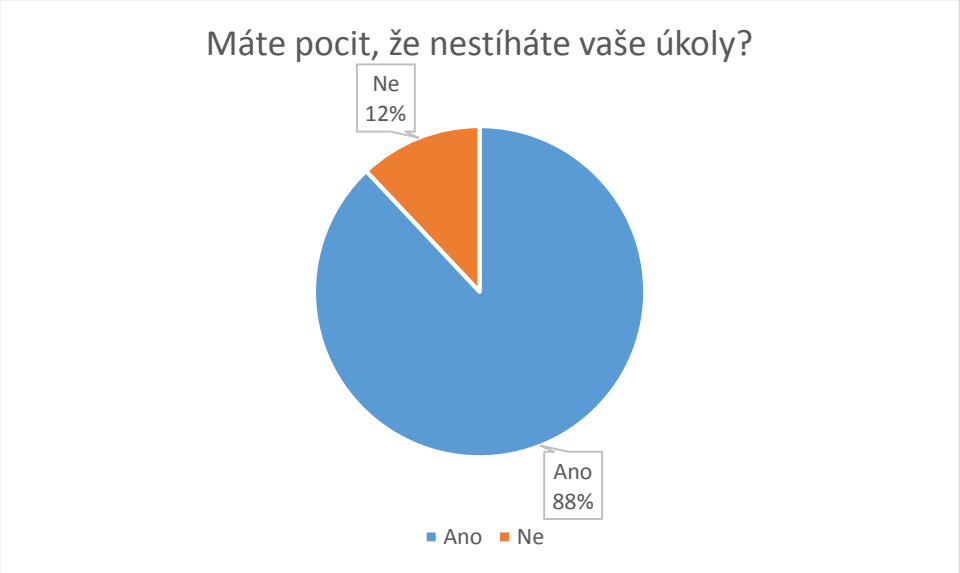
# 5 PŘEDCHÁZENÍ RIZIKŮM

Jedním z cílů mé aplikace je také předcházet rizikům při plánování projektů a plnění úkolů. Ať už se jedná o úkoly, které si stanoví sám uživatel, nebo o úkoly, které mu naplánuje vedoucí jeho projektu, vždy jejich nesplnění představuje riziko, kterému je lepší se vyhnout.

Abych zjistil, jak se k problému dodržování termínů zadaných úkolů staví moje testovací skupina uživatelů, vytvořil jsem dotazník, který měl za cíl zmapovat názory mé testovací skupiny. Dotazníku se zúčastnilo 51 uživatelů a odpovídali na tři následující otázky:

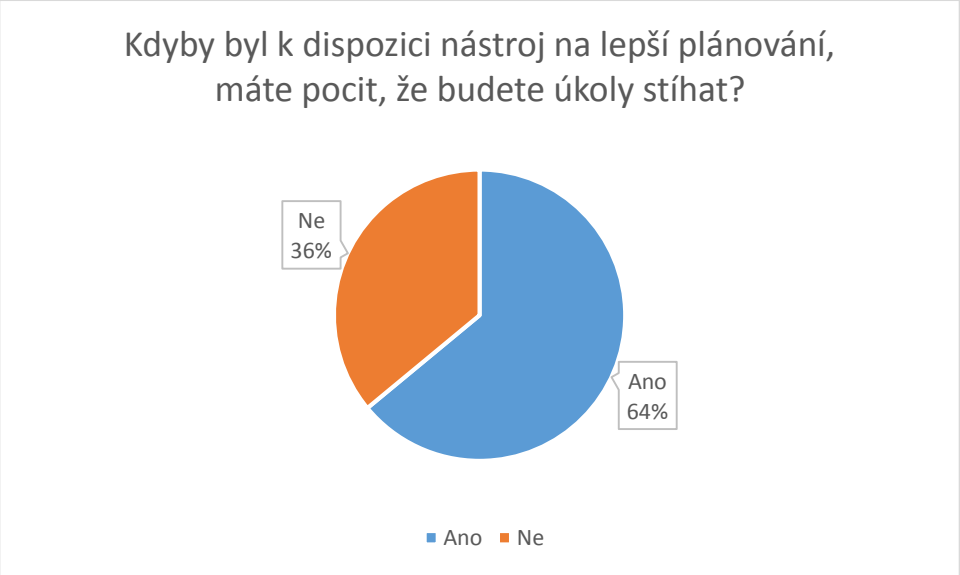
1. Máte celkově pocit, že nestíháte zadané úkoly?
2. Kdyby byl k dispozici nástroj, který by Vám umožnil úkoly lépe naplánovat, věříte, že je budete stíhat lépe?
3. Představte si, že máte zadáno 10 stejně těžkých úkolů, kolik jich podle vašich dosavadních zkušeností nestihnete do termínu?

Z grafu (Obrázek 5.1) je patrné, že velmi velké procento respondentů má problém se stíháním termínů a celkově převládá názor, že zadané úkoly spíše nestíhají.




*Obrázek 5.1 Odpovědi na první otázku*

V tomto okamžiku byla respondentům položena druhá otázka, a jak demonstruje graf (Obrázek 5.2), tak více než polovina si myslí, že vhodný nástroj by jim umožnil plnit úkoly v jejich termínech lépe, než tomu je dosud.




*Obrázek 5.2 Odpovědi na druhou otázku*


*Obrázek 5.3 Odpovědi na třetí otázku*

Poslední otázka měla sloužit k upřesnění otázky první, protože pocit, že uživatel úkoly nestihá je značně subjektivní. Velmi pečlivý člověk si bude už při jednom z deseti nestihnutých úkolů myslet, že nic nestihá, kdežto uživatel s laxním přístupem k dodržování termínů jich klidně poruší víc a stále bude mít pocit, že celkově vše stíhá.

Jak se ukazuje z grafu (Obrázek 5.3), tak většina respondentů patří k té poctivější části, nejvíc z nich by nestihlo pouze jeden hypotetický úkol a většina nepřekročí hranici čtyř úkolů z deseti.

Z výzkumu vyplývá, že dodržování termínů je problém, který je společný pro většinu mých testovacích uživatelů a v mé aplikaci se budu snažit tento problém vyřešit. Doufám, že tímto řešením se mi podaří snížit možné riziko negativního dopadu při nesplnění termínů zadaných úkolů.

# 6 NÁVRH APLIKACE

Jelikož je implementačním jazykem PHP a tento jazyk plně podporuje objekty, třídy a práci s nimi, bude celý můj návrh objektový a bude se snažit maximálně využívat dědičnost. Vzhledem ke zvolenému frameworku, který velmi aktivně podporuje použití návrhového vzoru MVP, jsem se rozhodl rozdělit návrh na presentery a modely. Slůvkem presenter je v Nette označována část controller v MVC, takže v případě Nette hovoříme o návrhovém vzoru MVP, tedy Model-View-Presenter (16). V této kapitole popíši pouze jednotlivé presentery, protože ve své aplikaci jsem zavedl konvenci, kde ke každému presenteru patří model stejného názvu, který se stará o manipulaci s daty, které danému presenteru přísluší.

## 6.1 PRESENTERY

Všechny presentery jsou v adresářové struktuře umístěny v adresáři presenters a všechny jsou předkem základního presenteru, který se v mé aplikace nazývá BasePresenter. Funkčnost, která je společná pro všechny presentery mé aplikace bude umístěna právě v tomto souboru.

### 6.1.1 Error presenter

Tento presenter se stará o zobrazování veškerých chyb v aplikaci, které nemají žádnou spojitost s aplikační logikou. Nepoužívá se tedy například pro výpis chyby, že nelze vytvořit novou událost. Jeho použití tkví v zobrazování chyb, které vrací webový server, tedy například chyba 404 – stránka neexistuje, nebo 500 – Chyba serveru. Pro každou tuto chybu je v adresáři obsahující view vytvořena šablona.

### 6.1.2 Events presenter

Events presenter má na starosti veškerou práci s typy událostí, které si může uživatel v mé aplikaci definovat. Obsahuje akce pro mazání, kopírování a sdílení typů událostí a definici formulářových komponent pro vytvoření nových typů událostí.

### 6.1.3 Homepage presenter

V budoucnu bude tento presenter použit pro zobrazení stránky, která je dostupná bez přihlášení a bude popisovat celý projekt. Aktuálně je ale aplikace po spuštění přesměrována na přihlašovací formulář, proto je zde tento presenter zatím nevyužit.

### 6.1.4 Projects presenter

Stará se o kompletní správu uživatelských projektů od jejich vytváření, sdílení, duplikování a výpis a mazání jednotlivých účastníků projektu.

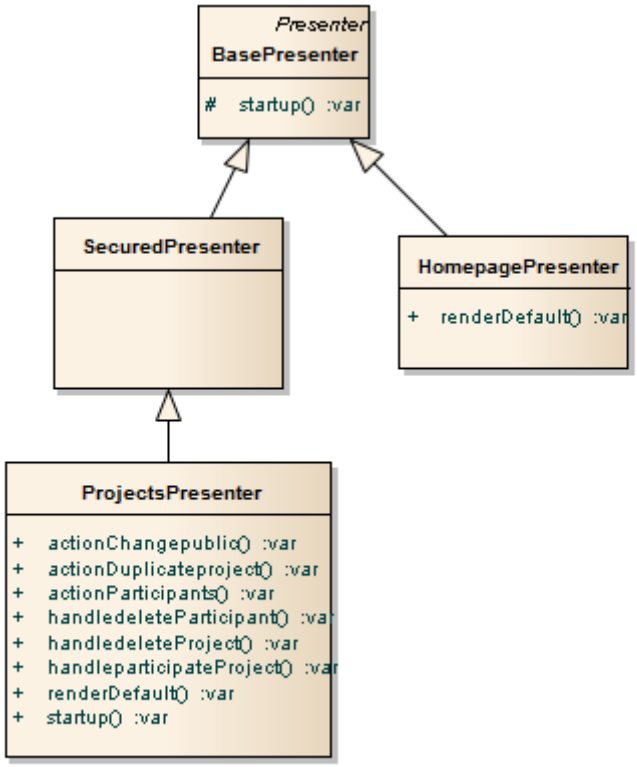
### 6.1.5 Secured presenter

Tento presenter se stará o přihlašování uživatelů a jejich registraci, pokud navštěvují web poprvé. Všechny presentery v aplikaci kromě Base a Homepage dědí od tohoto presenteru. Důvodem je jeho metoda startup, která se spustí vždy, když je k presenteru přistupováno a kontroluje, zda je uživatel přihlášen. Jakmile teda dojde ke spuštění Secured presenteru, nebo kteréhokoliv jeho potomka, dojde ke spuštění této metody a provede se kontrola. Díky dědičnosti je tak velmi elegantně vyřešen problém s bezpečností, kde se nemusím v každé metodě, která vyžaduje přihlášeného uživatele, ptát, jestli je opravdu přihlášen a případně ho přesměrovat na přihlašovací formulář. Vše co musím udělat pro vynucení přihlášení při přístupu k nějakému presenteru je nechat ho dědit od Secured presenteru. Na Projects presenteru demonstruje tento postup obrázek 7.1.

Secured presenter se dále stará o implementaci přihlášení přes účet u firmy Google, ale tomuto tématu budu věnovat samostatnou kapitolu (6.2).

### 6.1.6 Timeline presenter

Hlavní náplň celé aplikace, časové osy, má na starosti právě tento presenter. Stará se o vytváření časových os a událostí do nich patřících, obstarává různé druhy výpisů těchto os (jednotlivá osa, nebo všechny osy v projektu) a řeší práci s událostmi, jako je označování hotové práce, zamykání událostí a jejich úprava.




*Obrázek 6.1 Ukázka dědičnosti od Base presenteru*

## 6.2 GOOGLE PŘIHLAŠOVÁNÍ

Společnost Google, stejně jako mnoho jiných služeb, využívá k přihlašování do svých stránek otevřený protokol známý pod názvem OAuth (17). Tento protokol vznikl v roce 2006 jako reakce na standard OpenID (18), který umožňuje přihlašování do více služeb pomocí jednoho účtu. OpenID bohužel žádným způsobem neřeší oprávnění ke zdrojům, ke kterým má uživatel po přihlášení přístup, kdežto OAuth tento problém odstraňuje. V současné době už je dostupná i druhá verze, která nese název OAuth 2 a tuto budu ve své práci používat.

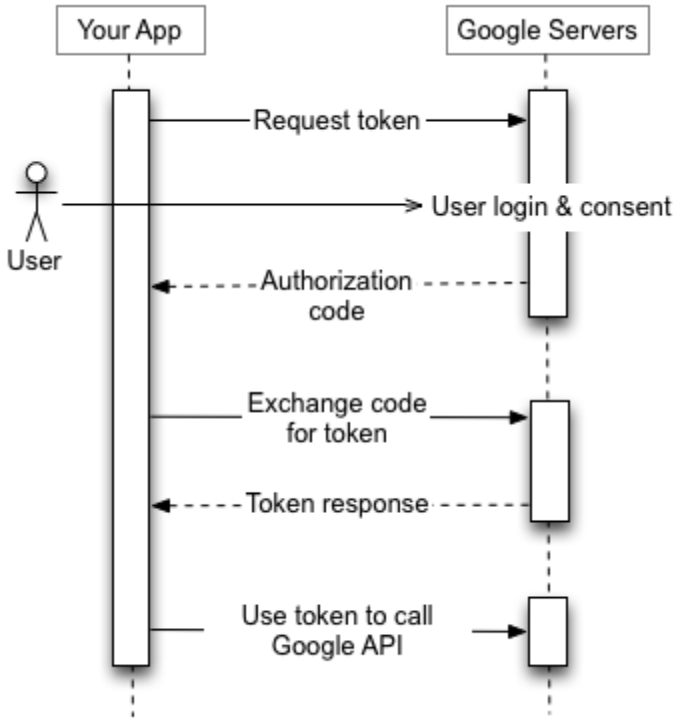
### 6.2.1 Princip funkce OAuth 2

Základním principem přihlašování přes cizí službu je fakt, že aplikace, které se přihlašuje, obvykle nedostává do rukou uživatelské heslo a jiné další citlivé údaje (pokud o ně nepožádá a uživatel s tímto nesouhlasí). Pokud se vývojář aplikace rozhodně přihlašovací službě důvěřovat, pak může brát identifikátor uživatele, který jeho aplikace po přihlášení

dostane, jako důvěryhodný. V mém případě se jedná o přihlášení pomocí účtu u společnosti Google.

Před samotným přihlášením dojde k definování zdrojů, ke kterým si moje aplikace přeje přístup a následný dotaz na přihlašovací službu už obsahuje kromě identifikátorů aplikace i požadavek na konkrétní zdroje.

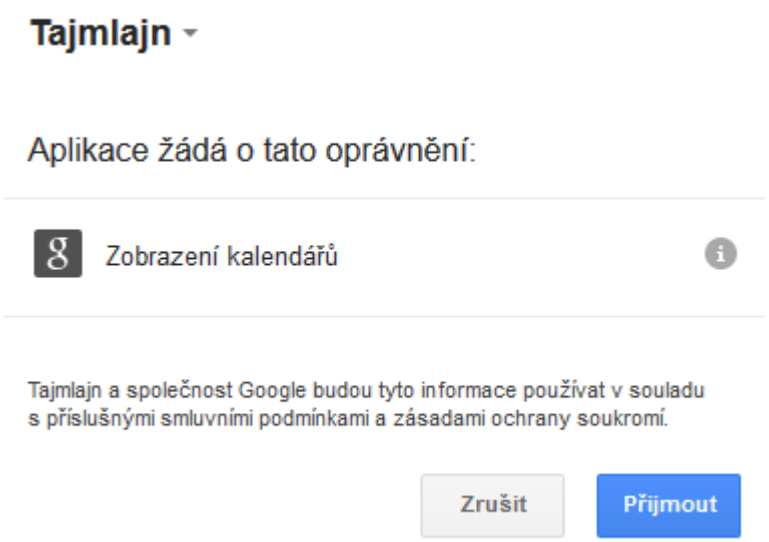
Princip fungování popisuje následující obrázek, který jsem převzal z dokumentace OAuth 2 na stránkách společnosti Google (19).




*Obrázek 6.2 OAuth 2 přihlašování*

Prvním krokem je dotaz mojí aplikace na přihlašovací službu, kam odesílám identifikátor své aplikace a jaké zdroje aplikace požaduje. Pokud identifikátor mé aplikace není platný, tak proces přihlášení tímto krokem končí. V případě přihlašování na Google je potřeba první zaregistrovat aplikace do speciálního přihlašovacího API, kde se vyplňují URL adresy aplikace a několik dalších informací. Na základě tohoto dostane aplikace přidělený jednoznačný kód a tajné heslo, které musí odeslat spolu s požadavkem na přihlášení.

Druhým krokem je pak přihlášení uživatele, kdy je jeho prohlížeč přesměrován na stránky poskytovatele přihlašovací služby, kde se přihlásí a potvrdí, že dává souhlas aplikaci s přístupem k požadovaným zdrojům (Obrázek 6.3)




*Obrázek 6.3 Potvrzení požadovaných zdrojů*

Uživatel je následně přesměrován zpět do aplikace a spolu s ním dostane aplikace autorizační kód. Jako třetí krok je nutné, aby samotná aplikace zadala požadavek přihlašovací službě na přístupový token. Pokud je autorizační kód správný, dostane aplikace přístupový token a tento pak může použít na volání dalších služeb poskytovatele přihlašování. Každý token je svázán na straně poskytovatele s konkrétním uživatelem a má pouze omezenou životnost (obvykle 30 minut), po které je potřeba token revalidovat pomocí revalidačního tokenu.

Při definici zdrojů, ke kterým chce mít aplikace přístup, je také možné definovat, jestli je pro přístup k nim potřeba aktivní přihlášený uživatel, nebo jestli může aplikace provádět požadavky jen na základě uloženého autorizačního tokenu. Toto se hodí například pro synchronizaci Google kalendáře, kdy se může tento proces provádět periodicky na pozadí a nenutí tak uživatele čekat dlouhou dobu po přihlášení na dokončení synchronizace.

Pozorný čtenář by se jistě mohl ptát, proč je zapotřebí výměna autorizačního kódu za token, proč se tento token nemůže předat aplikaci, když je uživatel přesměrován zpět do původní aplikace. Důvodem je bezpečnost, protože OAuth 2 se snaží uživatele maximálně chránit, tak se token vůbec nedostane do jeho prohlížeče, který by mohl být potencionálně

kompromitován. Prohlížeč dostane pouze autorizační kód, který je následně předán aplikaci a ta znovu provede dotaz na přihlašovací službu. I když v tomto kroku dojde například k odposlechnutí autorizačního kódu, útočník nemůže snadno požádat jménem aplikace přihlašovací službu o token, protože spolu s žádosti o něj se zasílá i identifikace aplikace a její tajné heslo a token je vrácen pouze na URL, které je v seznamu povolených URL adres pro přesměrování po přihlášení. Tato URL se definují na stránkách společnosti Google (na stejném místě, kde se registruje nová aplikace) a je potřeba být majitelem aplikace, aby tato URL šla změnit.

Z hlediska bezpečnosti jsou dnes tyto přihlašovací služby jedním z nejlepších řešení, jelikož pracují díky ověřeným metodám a úroveň zabezpečení je zde velmi vysoká. Nespornou výhodou pro uživatele je také možnost používat jeden účet pro přihlášení k více službám a nemuset si pro každou zakládat nový účet a heslo. Zároveň může využívat benefitů, které mu jednotný učet poskytuje (v případě mé aplikace je to například přístup ke Google kalendářům).

## 6.3 RESPONSIVNÍ DESIGN

Vzhledem k tomu, že jedním z požadavků při analýze byla i možnost využití na mobilních zařízeních, rozhodl jsem se od začátku vytvořit svou aplikaci jako responsivní. V tomto případě můžeme slůvko responsivní přeložit jako „přizpůsobivá“, protože aplikace se sama uzpůsobí šířce zařízení, na kterém je zobrazena, až do určitého limitu. Při návrhu responsivního designu musíme dodržet tři základní pravidla:

1. Použití relativních jednotek
2. Žádný obrázek nesmí mít pevnou velikost, ale musí se přizpůsobit velikosti stránky
3. Využití Media Queries

Použití relativních jednotek je nutné, protože jakožto designér aplikace neznáte předem velikost displeje, na kterém bude vaše stránka zobrazena. Definování šířky formuláře na pevných 400px může pro rozlišení 1920x1080 znamenat relativně malý formulář a pro mobilní telefon s rozlišením 800x480 to může být v landscape módu (na šířku) formulář přes celý displej. Trikem je definovat veškeré rozměry pomocí relativních jednotek, ideálně procent, která mají co největší počet desetinných míst. Procentuální jednotky se odvozují od elementu,

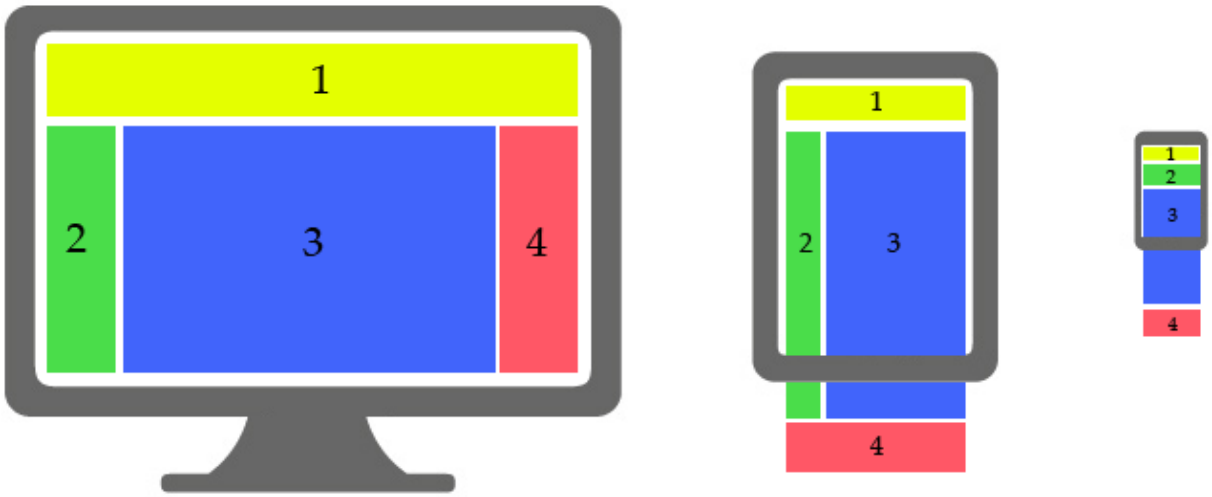
který je z hlediska zanoření HTML elementů nejblíž a má pevně definovanou velikost. Pokud žádný takový není, pak je tímto elementem okno prohlížeče.

Dalším bodem je přizpůsobení velikosti obrázků. Webový prohlížeč umí měnit velikost obrázku relativně snadno, ale pouze směrem dolů, tedy při zmenšování rozlišení. Při snaze ho zvětšit už není z čeho dopočítat obsah obrázku a dochází k výraznému snížení kvality (hovoříme o rastrovém obrázku, u vektorových tento problém samozřejmě nenastává). Cílem tedy je dovolit prohlížeči obrázek pouze zmenšovat, a jakmile šířka okna přesáhne originální šířku obrázku, tak ho už dále nemodifikovat. Tohoto se dá docílit pomocí definování výšky obrázku na automatickou a maximální šířku na 100% pomocí CSS.

Posledním bodem je využití Media Queries, což je součást standardu CSS3 a umožňuje aplikovat CSS pravidla na základě podmínek, které se vážou na zařízení, na kterém je stránka zobrazena. Designér tak může snadno definovat, že pro zařízení, které má rozlišení menší než například 600pxse budou mít některé prvky jiné styly, případně vůbec nebudou vidět. Typickým příkladem je velký obrázek v hlavičce webu, který na Full HD monitoru vypadá dobře, ale uživatel mobilního telefonu si ho neužije a při použití mobilních dat se počítá každý megabyte, takže stahovat obrovskou hlavičku, kterou na mobilním telefonu stejně nejde rozumně zobrazit, uživatele jen popudí.

Následující obrázek 7.4 ukazuje, jak by ideálně měl responsivní design fungovat. Vidíme pomocí čísel označené jednotlivé prvky webu, tedy hlavičku (1), menu (2), hlavní obsah (3) a boční panel (4) a jejich umístění. Můžeme si povšimnout, že jak se snižuje rozlišení, tak se prvky řadí za sebe tak, aby dávaly smysl. Například menu by mělo být vždy nahoře a ne až pod obsahem, aby ho uživatel nemusel složitě hledat.

Pro více informací o responsivním designu doporučuji skvělou knihu *Responsive Web Design* od Ethana Marcotte (20), který je považován za autora hlavní myšlenky responsivního designu a jeho kniha je jedna z prvních, která byla na toto téma napsána.




*Obrázek 6.4 Ukázka responsivního designu na ruzných typech zařízení*

# 7 IMPLEMENTACE A TESTOVÁNÍ

V této kapitole bych se rád věnoval popisu jednotlivých etap vývoje mé diplomové práce. Na začátku každé vývojové etapy mám požadavky, které chci v dané etapě splnit, následně je implementuji a předám uživatelům k testování. Z testování získám zpětnou vazbu, kterou následně použiji pro úpravu aplikace a vytyčení směru pro budoucí vývoj v dalších etapách.

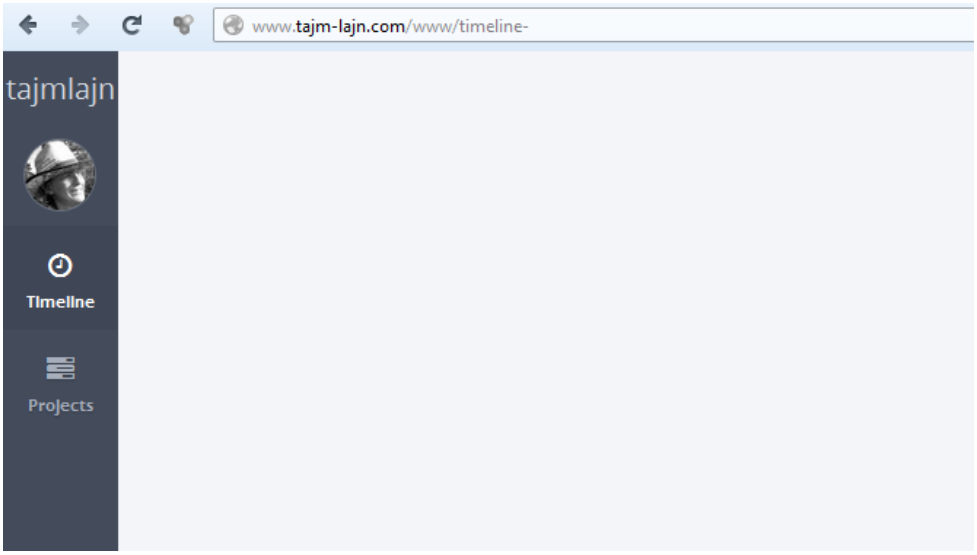
Kapitoly etap jsou rozděleny tak, že v hlavní kapitole etapy se věnuji samotné implementaci, co jsem udělal a jakým způsobem. V podkapitole testování zhodnotím, jak dopadlo testování s uživateli, u kterého jsem byl obvykle osobně přítomen a mohl pozorovat, jak uživatelé provádí jednotlivé akce. Podkapitola dotazník ukazuje výsledky dotazníku, který byl položen všem uživatelům, kteří aplikace v dané etapě používali.

V této kapitole budu velmi často používat termín „časová osa“, ačkoliv jeho použití je v mém případě značně nepřesné. Časová osa se váže na reálný čas a události mají velikost podle toho, kolik času reálně zaberou. V případě mé aplikace se spíše než o časovou osu jedná o seznam událostí seřazený podle času, kde nejnovější události jsou nahoře, ale z důvodu zjednodušení psaní textu budu tomuto zobrazení říkat časová osa.

## 7.1 PRVNÍ ETAPA

S touto etapou jsem začal už v období srpna 2013, snahou bylo vytvořit velmi jednoduchou kostru projektu, kde už bude použitý zvolený framework a všechny požadované technologie budou funkční, včetně možnosti přihlásit se pomocí účtu od společnosti Google. Přihlašování přes účet na sociální síti Facebook jsem přesunul do pozdějších etap, protože na rozdíl od přihlášení přes Google vyžaduje přihlášení přes Facebook vlastní aplikaci, která se u společnosti Facebook musí speciálně registrovat. Mým cílem bylo vytvořit prototyp co nejdřív, proto jsem implementoval pouze přihlašování přes Google účet, který vlastní všichni moji testovací zákazníci (obvykle se jedná o respondenty mého dotazníku).

V této fázi také bylo důležité vybrat konečný název mé aplikace. Nakonec jsem si vzal inspiraci z originálního projektu, ze kterého vycházím a aplikaci jsem pojmenoval „Tajm-lajn“ a zakoupil pro ni doménu http://www.tajm-lajn.com na které zveřejňuji nejnovější verze.




*Obrázek 7.1 Základní kostra aplikace Tajm-lajn*

### 7.1.1 Testování

Jelikož cílem této etapy bylo vytvoření kostry, která půjde nahrát na web a ve které bude fungovat pouze základní přihlašování přes Google účet, tak jsem do testování nezapojoval všechny účastníky mého dotazníku, ale pouze pár nejbližších přátel. Výsledkem této etapy je otestovaná kostra aplikace, která obsahuje základní vzhled a funkční přihlašování.

## 7.2 DRUHÁ ETAPA

V druhé vývojové etapě už jsem si dal za cíl vytvořit první použitelnou aplikaci, která sice bude mít omezené možnosti, ale už bude umožňovat plánovat čas. Po přihlášení by měl mít uživatel dostupnou časovou osu, do které si může plánovat vlastní události. Časová osa by měla být přehledně graficky znázorněna, aby bylo na první pohled patrné, co se dělo kdy.

U každé události musí jít definovat její název, popisek, typ a datum, kdy se má vykonat. Typy událostí jsem prozatím definoval napevno jako Práce, Telefonát a Schůzka. Do budoucna jistě plánuji možnost přidávat vlastní typ události, ale tyto tři jsou dle mého názoru nejčastější při plánování času. Tuto informaci jsem získal tak, že jsem prošel svůj pracovní kalendář za poslední rok a pozoroval, jaké události nejčastěji plánuji.


*Obrázek 7.2 Časová osa s událostmi*

### 7.2.1 Testování

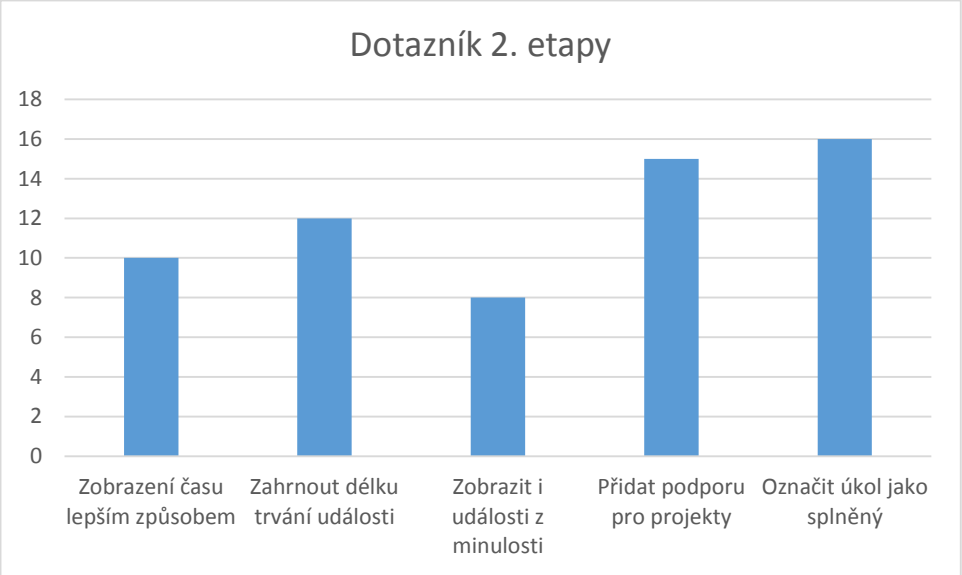
Po implementaci jsem předal výsledný produkt na otestování větší množině uživatelů, abych hned od začátku zjistil, jestli jde vývoj správným směrem, nebo jestli jsou moje představy o intuitivním uživatelském rozhraní mylné.

Výsledkem testování bylo několik důležitých poznatků, které jsem se rozhodl zahrnout do specifikace pro třetí vývojovou etapu. Tohoto testování se zúčastnilo kolem 20 uživatelů a v této fázi zatím neřeším kompatibilitu mezi prohlížeči, takže bylo testerům doporučeno použití webového prohlížeče Mozilla Firefox, nicméně aplikace bez problémů funguje i v prohlížeči Google Chrome.

Část testování zahrnovala také pozorování uživatelů při práci a sledování, kam nejčastěji klikají myší a případně ve které části aplikace se nejvíc ztrácí, nebo co se snaží dělat. Výsledky tohoto pozorování se následně promítnou do dalších vývojových etap mé aplikace.

### 7.2.2 Dotazník

Po otestování druhé etapy jsem vytvořil dotazník, co by uživatelé uvítali dál, případně o čem si už teď myslí, že je zbytečné. Odpovědi na první otázku sumarizuje graf v následující kapitole, v případě druhé otázky jsem nedostal žádnou odpověď. Soudím, že v aplikaci je toho zatím příliš málo, než aby měl uživatel pocit, že obsahuje něco, co nevyužije. Podobnou sadu otázek hodlám předkládat testovací skupině po každé vývojové etapě.




*Obrázek 7.3 Dotazník 2. etapy*

## 7.3 TŘETÍ ETAPA

Z testování druhé etapy vzešlo mnoho podnětů pro budoucí vývoj aplikace. Tento fakt je pochopitelný, vzhledem k tomu, že se uživatelům poprvé dostala do ruky verze programu, která umí i něco víc, než jen přihlásit se přes Google účet. Sesbíral jsem velké množství podnětů a podobně jako u dotazníku vybral ty nejčastější, u kterých předpokládám, že pokud se na nich shodne většina uživatelů, tak pro mě budou opravdu důležité. Nejčastější podněty jsou zobrazeny v následujícím grafu.

Další podněty jsem se rozhodl prozatím odložit stranou a implementovat tento seznam požadavků a na základě dalšího testování určit priority. Zároveň v této fázi zjišťuji, že každý uživatel by chtěl od aplikace něco trochu jiného a tak bude nalezení, které bude vyhovovat většině uživatelů náročnější, než jsem zprvu předpokládal.


*Obrázek 7.4 Časová osa po implementaci třetí etapy*

Jak je patrné z obrázku 8.4, tak došlo ke značným změnám uživatelského rozhraní, každá událost typu Práce (modrá ikonka špičatých závorek) má barevný nadpis podle stavu, červený pro nehotovou a zelený pro hotovou. Také čas události doznal jisté změny. Pokud je v horizontu týdne, tak místo data obsahuje den, kdy událost nastane a hodinu.

Změnil jsem také řazení událostí, nyní mám dva druhy událostí, budoucí a minulé. Zatímco budoucí jsou řazeny tak, že čím je událost blíž zelenému nápis s názvem časové osy, tím blíž je současnosti, tak události minulé jsou řazeny až pod událostmi budoucími. Zatím nevím, jestli tento způsob řazení je právě to nejlepší řešení, ale zdálo se mi logické, že uživatel jako první chce vidět události, které ho čekají a až pokud chce vidět minulé události, tak musí projít všechny budoucí.

Zároveň jsem implementoval novou část systému, která umožňuje přidávat vlastní projekty a do těchto projektů časové osy. Projekt a časová osa nesou informaci o svém názvu a popisku. Spolu se zavedením projektů byla dána uživatelům možnost označit projekt jako veřejný. Jakmile je projekt evidován jako veřejný, mají ostatní uživatelé aplikace možnost naklonovat si projekt. Klonováním zde rozumíme kopii projektu včetně názvu a časových os uvnitř.

### 7.3.1 Testování

Testování této etapy se zúčastnilo kolem 30 uživatelů, což je více než u druhé etapy a výsledků jsou proto rozmanitější. Ukázalo se, že současný systém zobrazování asi nebude úplně nejvhodnější. Každý uživatel dostal pokyn, aby si zkusil vytvořit projekt Diplomová práce a v něm si nadefinovat několik časových os. Výsledkem bylo, že téměř každý měl ve svém projektu časové osy pojmenované „první týden“, „druhý týden“ atd. Podle mého názoru (a zpětné vazby uživatelů) by toto měla aplikace nativně podporovat, než nutit uživatele přemýšlet, který týden vývoje je teď. Další problém nastává při pohledu do minulosti, kdy pro nalezení konkrétní události, u které uživatel nezná přesné datum, musí zbytečně projít velké množství časových os.

Dalším požadavkem byla možnost sdílet projekty mezi uživateli, protože například pro projekt stylu diplomové práce mají obecný základ všichni uživatelé stejný (termíny pro odevzdání práce, konzultace). Proto by bylo dobré v další etapě myslet na tuto funkčnost.

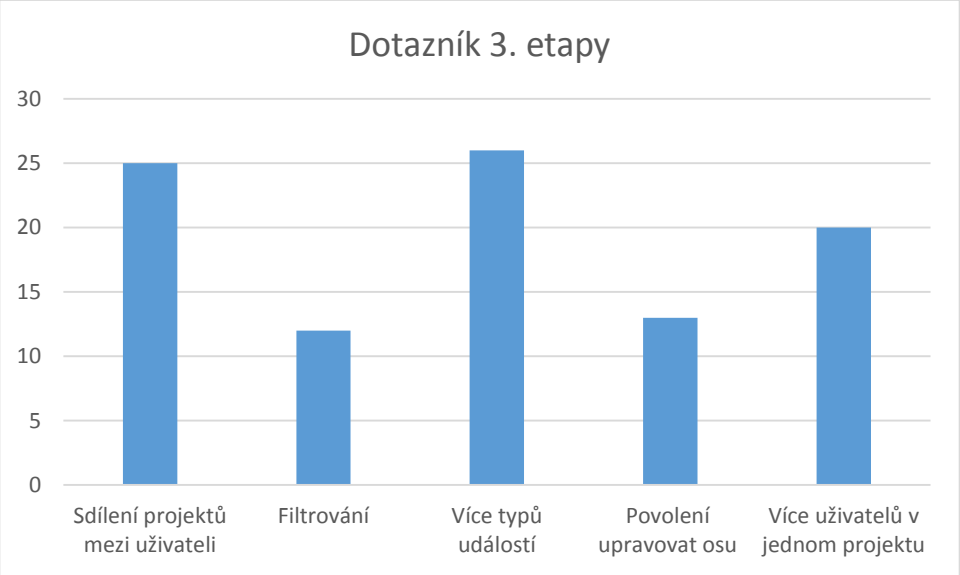
Nevýhodou aktuálního zobrazení časové osy je fakt, že jsem při jejím návrhu naprosto ignoroval rozměry zařízení, na kterých bude zobrazena. Dnes je téměř běžné mít širokoúhlý

monitor, který má na šířku téměř dvakrát tolik prostoru, co na výšku. Pokud chci efektivně zobrazit co největší množství událostí, musím předělat časovou osu z aktuálního vertikálního zobrazení na horizontální.

### 7.3.2 Dotazník

Výsledky tohoto dotazníku už jsou výrazně zajímavější, než v případě druhé etapy, kdy aplikace obsahovala ještě velmi málo aktivních prvků. Stejně jako v minulých dotaznících i zde jsem položil uživatelům dvě otázky a to jakou funkčnost by uvítali v další etapě a jaké věci by se naopak měli odstranit, nebo předělat.

Výsledky první otázky zobrazuje následující graf:




*Obrázek 7.5 Dotazník 3. etapy*

Na druhou otázku bylo k dispozici textové pole, kam se mohl uživatel rozepsat, takže výsledek bude sumář odpovědí, kde nejčastěji se vyskytuje právě kritika zobrazení událostí na výšku a nevyužití prostoru na šířku. Druhá nejvíce zmiňovaná úprava bylo odstranění délky trvání události. Ukázalo se totiž, že pro účely zobrazení časové osy a celkový smysl aplikace je délka události nepotřebnou informací. Cílem je uživatele informovat, že má něco udělat, ale délku nechává na něm. V budoucích etapách bude délka události skryta, ale ponechána ve zdrojovém kódu aplikace pro případ, že by se k ní autor chtěl někdy později vrátit.

## 7.4 ČTVRTÁ ETAPA

Na základě testování v předchozí etapě se jako hlavní problém ukázalo zobrazení události na výšku a proto jsem chtěl tento problém odstranit jako první. Po prvotním předělání osy na horizontální zobrazení jsem ale zjistil, že výsledek je velmi nepřehledný. Události byly přesunuty nad a pod osu, jak zobrazuje následující obrázek:


*Obrázek 7.6 Horizontální časová osa*

Při měření počtu zobrazených událostí se na rozlišení 1920x1080 zobrazí pohodlně 7 událostí vertikálně a 7 událostí horizontálně. Výsledek je tedy naprosto stejný a důvodem je délka popisu události. V případě vertikálního zobrazení je k dispozici celá šířka monitoru na zobrazení textu události a ve většině případů tak mají události pouze jednořádkový popis (popis události obvykle není delší než dvě věty). Zatímco u vertikálního zobrazení tak s délkou není problém, v případě horizontálního už je délka popisu důležitým faktorem. Pokud bude pole pro událost příliš malé, nebude zobrazení přehledné, a i když se na plochu vejde více událostí, uživatel stráví delší čas hledáním, než v případě vertikálního zobrazení. Pokud naopak uděláme pole moc široké, pak nedojde ani k zobrazení všech 7 událostí. Pro představu, pole události na obrázku má šířku 200px.

Dalším problémem při horizontálním zobrazení bylo posouvání, jelikož zde došlo k použití horizontálního posuvníku. Na otázku „Na který posuvník jsou uživatelé webových stránek více zvyklí“ není, dle mého názoru, ani potřeba dělat dotazníkové šetření. Velké množství stránek dnes používá vertikální posuvník a technologie zobrazující webové stránky se tomuto trendu plně přizpůsobila. Kolečko na myši je vertikální a pohyb prstu koresponduje s pohybem webové stránky na monitoru. Horizontální posuvník oproti vertikálnímu nemá prakticky žádnou podporu, a pokud ho stránka obsahuje, pak jde většinou buď o chybu, nebo o designový prvek.

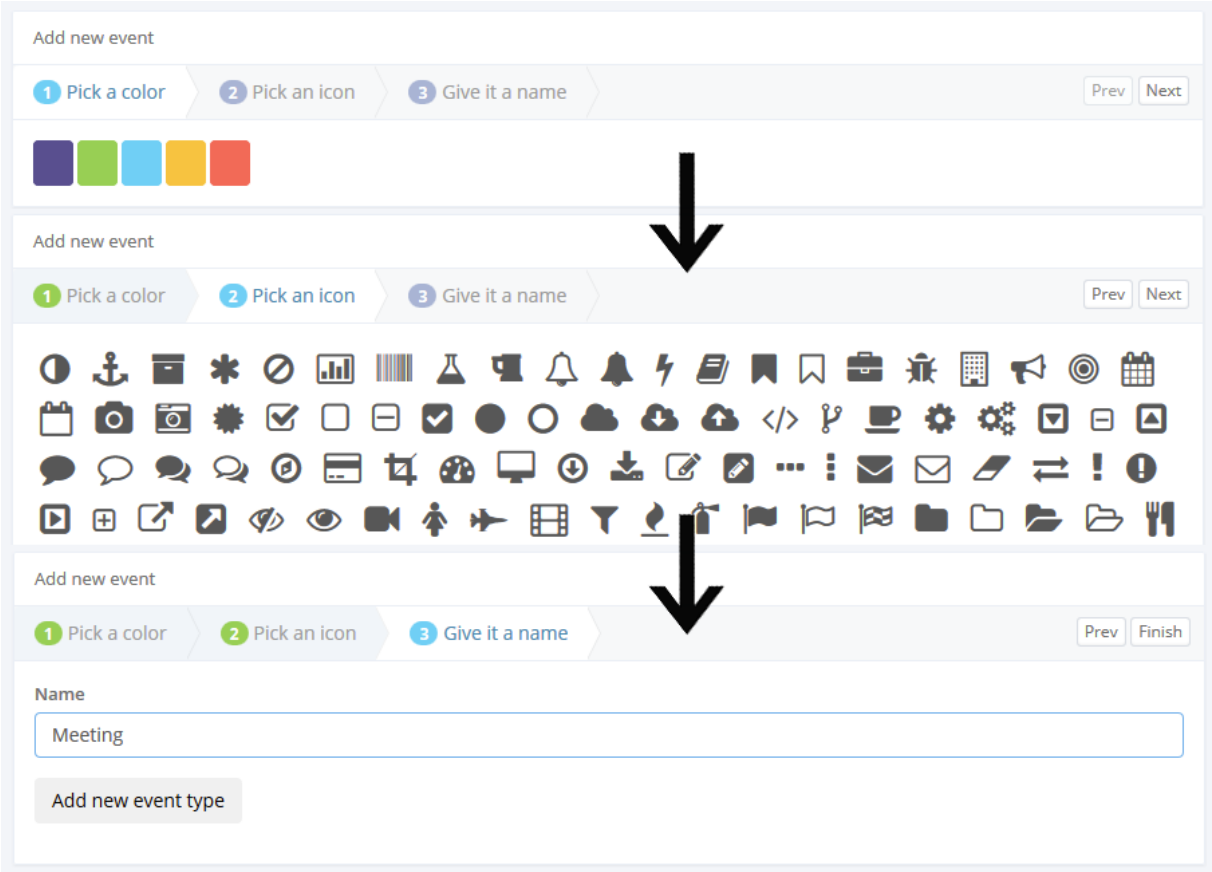
Snažil jsem se uživatelům přiblížit možnost využití horizontálního posunu tím, že při vertikálním posunu kolečkem na myši uvnitř časové osy došlo k posunu horizontálnímu. Tu samou funkčnost jsem implementoval i pro kurzorové klávesy. Požádal jsem uživatele, kteří v předchozím dotazníku žádali o změnu zobrazení, aby tuto funkcionalitu vyzkoušeli a výsledek byl katastrofální. Díky mohutnému rozšíření vertikálního posunu totiž není nikdo zvyklý na posun horizontální a tak se stalo uživatelské rozhraní velmi nepříjemné na používání. Uživatelé si stěžovali, že změna standardního chování myši a klávesnice není dobrá cesta, ale po druhém testu, kde pro posun používali horizontální posuvník v prohlížeči, na který museli kliknout myší a táhnout ho v požadovaném směru se ukázalo, že tímto směrem cesta také nevede.

Po několika dalších testech jsem usoudil, že horizontální zobrazení byla slepá ulička a časová osa tak v budoucích etapách vývoje bude opět vertikální. Jediný testovací uživatel, kterému se změna na horizontální osu zamlouvala, byl majitel myši s dotykovým povrchem, kde horizontální i vertikální posun dělá pouze přejížděním prstu po dotykové ploše v požadovaném směru. Pokud dojde v budoucnu k masivnějšímu rozšíření této technologie, mohu o přesunu na horizontální osu opět uvažovat.

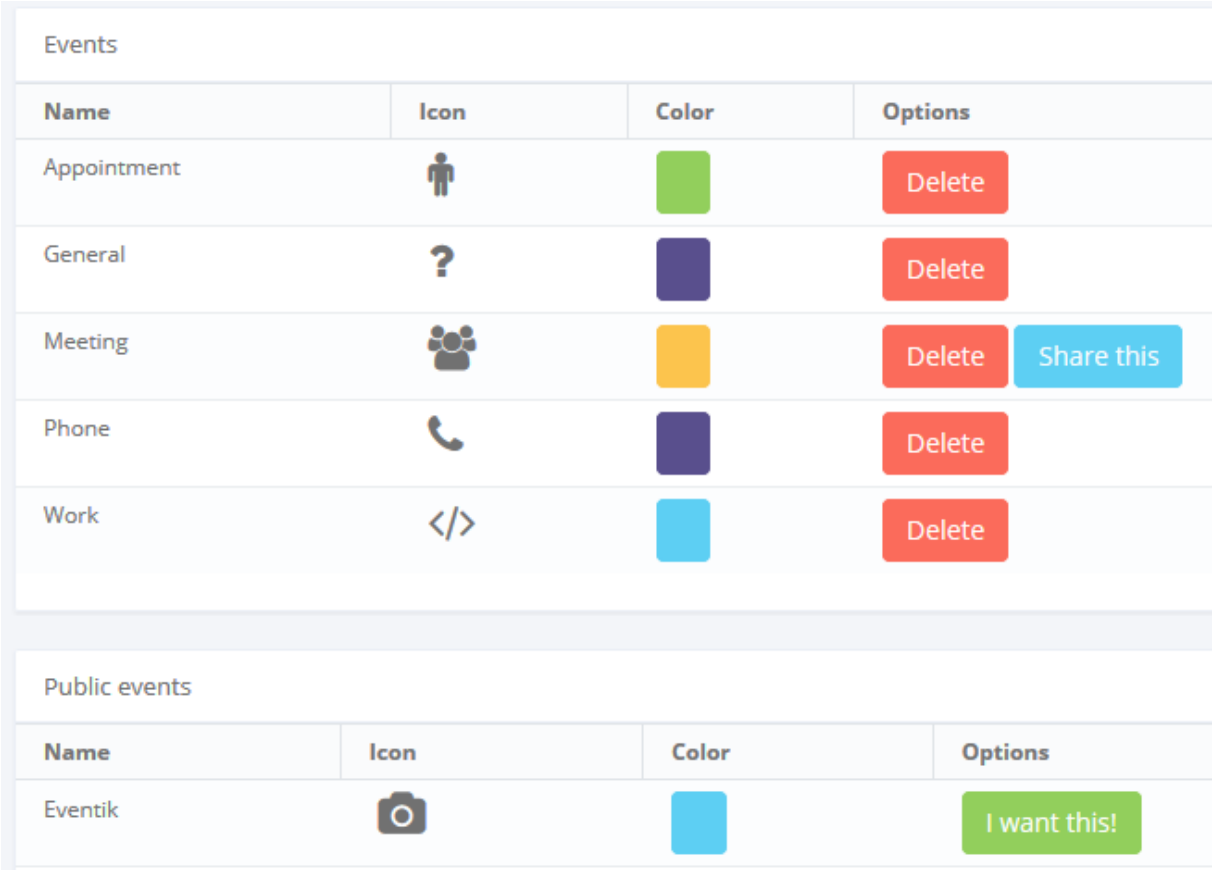
### 7.4.1 Typy událostí

Po definitivním rozhodnutí, jak bude vypadat zobrazení časové osy, jsem začal implementovat ostatní požadavky uživatelů podle množství odpovědí v dotazníku třetí etapy. Jako první se jedná o více typů událostí, protože dosud byly typy událostí definovány napevno a Telefonát, Práce a Schůzka. Pro reálné plánování je ale potřeba typů mnohem víc a tak bylo vytvořeno rozhraní pro vytváření vlastních typů událostí formou jednoduchého průvodce, kde si uživatel nejprve vybere barvu, následně ikonu a nakonec název události. Ve kterékoliv fázi průvodce se může uživatel vrátit k předchozímu kroku. Celý proces tvorby demonstruje obrázek 8.8.

Jelikož předpokládám, že některé typy události budou u velkého počtu uživatelů velmi podobné, tak došlo k přidání mechanismu pro sdílení těchto typů událostí. Po vytvoření nového typu události může uživatel přes seznam typů události sdílet pomocí jednoho kliknutí. Podobným způsobem může převzít typ někoho jiného. Seznam typů události a mechanismus sdílení je zobrazen na obrázku 8.7.




*Obrázek 7.8 Průvodce pro tvorbu nového typu události*



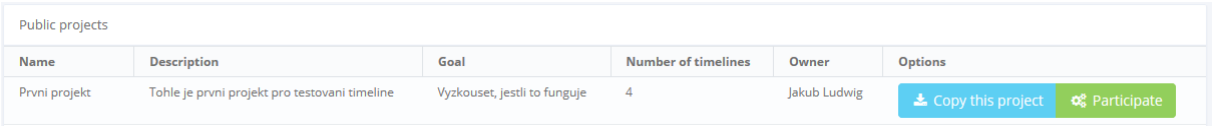

*Obrázek 7.7 Zobrazení typů událostí*

### 7.4.2 Správa projektů a časových os

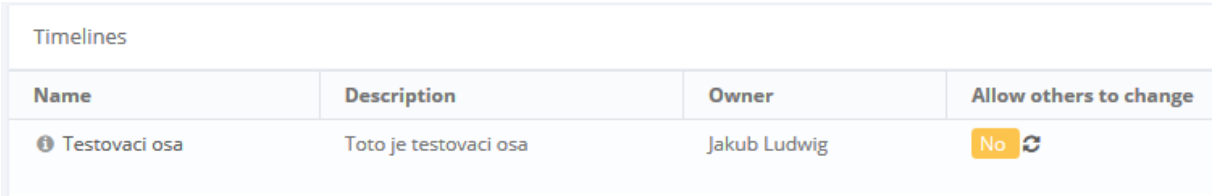
Dalším požadavkem uživatelů byla možnost sdílení projektů i jiným způsobem, než klonováním projektu. Aktuální mechanismus pouze kopíruje šablonu projektu, ale neumožňuje na projektu pracovat společně. Cílem je tedy vytvořit k projektům nástroj, který umožní pracovat na nich najednou spolu s dalšími uživateli.

Na základě tohoto podnětu bylo do administrace projektů přidáno k veřejným projektů tlačítko Participate, které umožňuje uživateli přidat se k projektu a vytvářet v něm vlastní časové osy. Požadavky na sdílení projektů mezi uživateli a více uživatelů v jednom projektu se tímto dají považovat za splněné, ale pro správu časových os je zde ještě jeden oprávněný požadavek a to možnost zamknout časovou osu proti úpravám ostatními uživateli v projektu. Požadavek se dá chápat i opačně, tedy dát vlastníkům jednotlivých časových os možnost povolit ostatním účastníkům projektu do nich zasahovat.

V praxi bude mít tato funkcionalita využití například při řízení malého projektu, kde každý člen bude mít vlastní osu a pak zde bude jedna společná, kam mohou přidávat události všichni.




*Obrázek 7.9 Zobrazení veřejného projektu*




*Obrázek 7.10 Možnost povolení změn v časové ose*

### 7.4.3 Filtrování

Posledním požadavkem na tuto vývojovou etapu byla implementace filtrování do zobrazení časové osy. Proto došlo k implementování filtrů na čas a typ události, které se zobrazují u časové osy. Přidání filtrů dává uživatelům možnost mít vytvořených méně časových os a lépe se v nich orientovat.

### 7.4.4 Testování

Jelikož v této etapě došlo k implementaci velkého množství nových prvků, tak jsem uznal za vhodné rozšířit testovací skupinu, čímž se osobního testování a dotazníku se v této fázi účastnilo dohromady 47 uživatelů.

Pro toto testování jsem uživatele rozdělil do dvojic a jednomu jsem zadal roli vedoucího projektu, kdy jeho cílem bylo vytvořit projekt a do něj jednu vlastní a jednu společnou časovou osu. Druhému uživateli byla přiřazena role kolegy prvního uživatele a jeho úkolem bylo přidat se do projektu, vytvořit svou vlastní osu a naplánovat pár událostí do společné osy. Oba uživatelé také měli za úkol vytvořit několik vlastních typů událostí.

Po pozorování, jak se daných rolí uživatelé zhostili, jsem zjistil, že má práce vůbec nebere v potaz, že by někdo mohl být vedoucí projektu. Takže uživatel se sice přidá k projektu, ale neexistuje zde možnost, jak ho odstranit a majitel projektu i takto přidaný uživatel mají naprosto stejná práva na projekt.

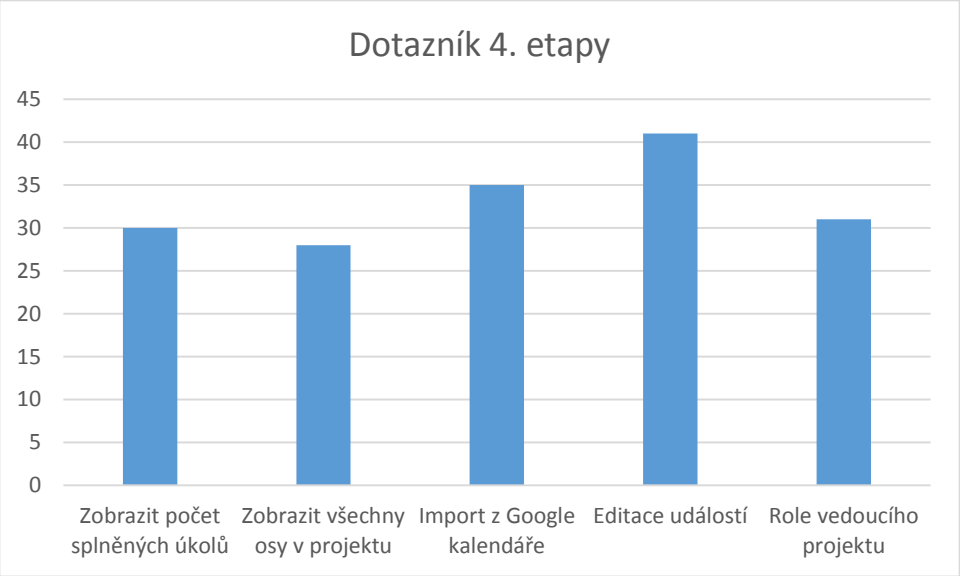
Dalším problémem, kterého jsem si všiml, je fakt, že vytvořené události už nejdou nijak upravovat, do budoucna tedy musím vyřešit úpravu událostí.

Poslední zjištění je bezpečnostního rázu, kdy aplikace do teď žádné zabezpečení neřešila, ale s příchodem více uživatelů došlo i na tento typ testování a tak se zjistilo, že pokud uživatel zná správné URL, může mazat projekty, u nichž není vlastníkem a velké množství dalších akcí, na které v mé aplikaci chybí kontrola. Před pátou etapou tak dojde k analýze rizik a vyhodnocení kritických chyb, které bude třeba opravit.

### 7.4.5 Dotazník

Jako v předchozích fázích i tentokrát jsem položil uživatelům dvě otázky, tedy co by rádi viděli v následující vývojové etapě, čeho bych se naopak měl zbavit, a co je v aplikaci podle jejich názoru zbytečné.

Odpověď na první otázku sumarizuje následující graf:




*Obrázek 7.11 Dotazník 4. etapy*

Jak si můžeme povšimnout, požadavky uživatelů se shodují s mým vlastním zjištěním po pozorování práce s aplikací. Možnost zobrazení všech os v projektu vzešla především od uživatelů, kteří měli roli vedoucího projektu, protože podle jejich tvrzení už při několika málo časových osách se stává projekt nepřehledným a při zobrazení jen jedné časové osy se naprosto ztrácí vazba na zbytek projektu a na práci ostatních kolegů. Většina vedoucích také při vytváření projektu otevřela svůj kalendář a začala přepisovat události z něj, protože se jim nechtělo vymýšlet nové, z čehož vzešel požadavek na import z Google kalendáře.

Z druhé otázky v této etapě nevzešel žádný smysluplný výstup, protože názory uživatelů byly značně rozdílné na to, co by aplikace měla a neměla obsahovat. Rozhodl jsem se proto, že pokud nezávisle na sobě neřekne alespoň 30% respondentů, že jim daná funkčnost vadí, nebudu se tímto názorem zabývat, protože názor každého uživatele je jiný.

## 7.5 ANALÝZA RIZIK ČTVRTÉ ETAPY

Jelikož testování na konci této etapy ukázalo, že aplikace obsahuje řadu bezpečnostních děr, rozhodl jsem se provést důkladnou analýzu rizik a pro další etapy odstranit ta rizika, která vyhodnotím jako kritická. Pro zjištění a ohodnocení rizik jsem se rozhodl použít metodu FMEA (Failure Mode and Effects Analysis) (21). V analýze předpokládáme pouze případy, kdy se na aplikaci snaží zaútočit někdo zvenčí a nebere do úvahy problémy hardwarové povahy, ani žádné další faktory, jako například živelné katastrofy.

### 7.5.1 Metoda FMEA

Tato metoda se řadí mezi kvalitativní metody analýzy rizik a jako první s ní přišla americká armáda, která ji používala na odhalení závad armádních strojů a zařízení. Později ji převzala NASA a následně i společnost Ford, kde se metoda FMEA uplatňovala při kontrole součástek vozidel. Jedná se o metodu, která je založena na otázce „Co se stane, když …“.

Principem FMEA je analyzovat systém a dekomponovat ho na menší části, kde o každé této části se uvažuje jako o samostatném systému a je snahou identifikovat, co vše se může porouchat, co je příčinou a jaké jsou následky případné poruchy. Výstupem metody je tabulka, která obsahuje jednotlivá rizika, příčiny jejich vzniku, závažnost dopadu a celkovou kritičnost daného scénáře. Následně se pomocí definované stupnice rozdělí rizika na běžná, závažná a kritická a v případě kritických pak musí dojít k jejich nápravě alespoň takovým způsobem, aby se přesunula do kategorie závažná, ideálně však do kategorie běžná.

### 7.5.2 Určení stupnic

V tomto případě budu pracovat pouze se třemi stupnicemi a to se stupnicí pravděpodobnosti výskytu rizika, dopadu rizika a kritičnosti rizika. Každá stupnice obsahuje jak číselnou hodnotu, tak slovní popis.

| Míra | Pravděpodobnost |
|---|---|
| Malá | 0 – 0,3 |
| Střední | 0,31 – 0,6 |
| Velká | 0,61 – 1 |

*Tabulka 1 Pravděpodobnost výskytu rizika*

| Dopad | Rozsah |
|---|---|
| Malá | 0 – 30 |
| Střední | 31 – 60 |
| Velká | 61 - 100 |

*Tabulka 2 Dopad rizika*

| Velikost | Rozsah |
|---|---|
| Běžná | 0 – 20 |
| Závažná | 21 – 40 |
| Kritická | 41 - 100 |

*Tabulka 3 Kritičnost rizika*

Rozsah pro kritičnosti rizika (Tabulka 3) je určen vynásobením horní hranice rozsahu pravděpodobnosti a dopadu.

### 7.5.3 Identifikace rizika

Jako jedno z hlavních rizik považuji podvrhnutí přihlášení uživatele, tedy situaci, kdy se bude moci útočník přihlásit za jiného uživatele bez jeho vědomí. Tato situace může vzniknout několika způsoby, chyba může být v OAuth přihlašování (R1), dále v neošetření SQL Injection ve formulářích (R2) nebo špatnou manipulací s proměnnou sezení (R3).

Dalším rizikem je bezesporu možnost manipulace s obsahem, ke kterému by útočník neměl mít přístup, ať už se jedná o vlastnictví projektu, časové osy, nebo konkrétní události. Tento útok může být realizován několika způsoby, mezi které řadíme změny parametrů v URL dotazu (R4), neošetření SQL Injection ve formulářích (R5), Cross-site scripting útok (R6), krádež proměnné sezení (R7) nebo nedostatečná implementace oprávnění, která útočníkovi dovolí například měnit hodnoty formulářových polí, které po odeslání způsobí změnu jiných zdrojů, než které vlastní (R8).

Takto identifikovaná rizika vložíme do tabulky a ke každému záznamu vložíme míru pravděpodobnosti výskytu tohoto rizika a závažnost jeho dopadu. Pro získání objektivního hodnocení je vhodné využít více odborníků na danou oblast a nezávisle na sobě je nechat

aplikaci analyzovat a rizika ohodnotit. Pro mé hodnocení se mi podařilo získat tři odborníky, kteří se buď přímo zabývají bezpečností v IT, nebo s ní jejich práce úzce souvisí. Z jejich ohodnocení jsem udělal aritmetický průměr a výsledkem je následující tabulka. Kompletní hodnocení každého odborníka je k nalezení v přílozePříloha A.

| Riziko | Pravděpodobnost | Dopad | Kritičnost |
|---|---|---|---|
| R1 | 0,1 | 82 | 8,2 |
| R2 | 0,5 | 80 | 40 |
| R3 | 0,35 | 79 | 27,65 |
| R4 | 0,8 | 60 | 48 |
| R5 | 0,5 | 78 | 39 |
| R6 | 0,4 | 42 | 16,8 |
| R7 | 0,3 | 70 | 21 |
| R8 | 0,8 | 62 | 49,6 |

*Tabulka 4 Sumář hodnocení rizik*

Na základě uvedených hodnot (Tabulka 4) jsem identifikoval 2 běžná rizika (R1 a R6), 4 závažná rizika (R2, R3, R5 a R7) a 2 kritická rizika (R4 a R8). Protože je v mém nejlepším zájmu mít aplikaci co nejbezpečnější, rozhodl jsem se vyřešit i závažná rizika. Následující tabulka (Tabulka 5) uvádí pro každé identifikované závažné a kritické riziko opatření, které provedu, abych buď snížil pravděpodobnost jeho výskytu, nebo dopad.

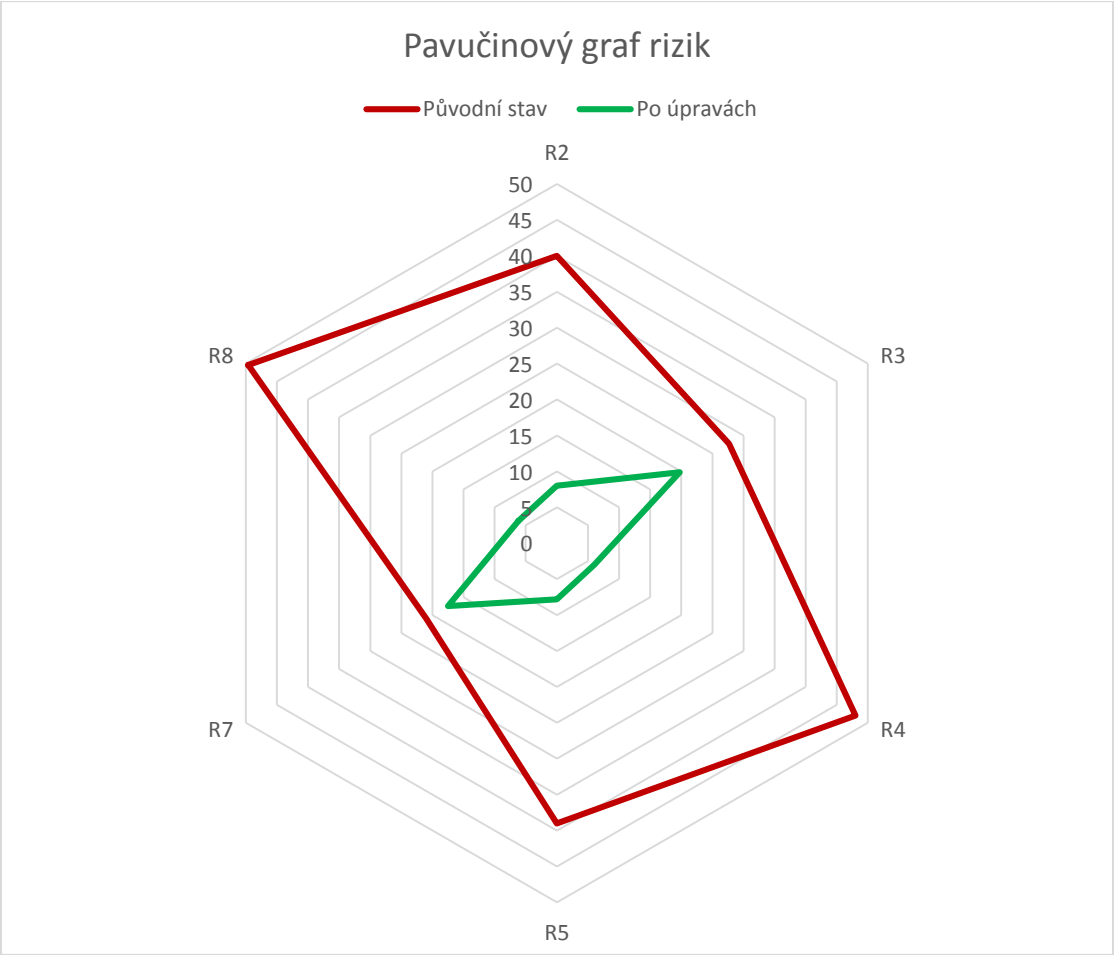
Rizika R2 a R5, tedy SQL Injection zranitelnost jsem se rozhodl vyřešit pomocí použití databázové vrstvy Dibi (22), které se sama stará o ošetření vstupních proměnných přidáním escape sekvencí před každý potenciálně nebezpečný znak. Rizika R3 a R7, tedy problém s proměnnou sezení jsem vyřešil použitím mechanismu (23) frameworku Nette pro práci s proměnnou sezení, namísto ruční práce s proměnnou $_SESSION. Framework se sám postará o zabezpečení proměnné sezení hned několika způsoby, mezi které patří například zákaz modifikace cookies pomocí javascriptu nebo pravidelné regenerování identifikace sezení (session_id).

Poslední dvě rizika R4 a R8 jsou vlastně jeden společný problém, který lze způsobit několika způsoby, ale výsledkem je vždy manipulace se zdroji, ke kterým nemá mít uživatel přístup. Proto jsem do každé funkce, které nějakým způsobem manipuluje s databází, přidal doplňující podmínku, která kontroluje, zda je přihlášený uživatel zároveň vlastníkem daného záznamu. Pokud není, pak se modifikace neprovede. Útočník tak může měnit parametry URL nebo skrytá pole ve formulářích, ale neměl by být schopen úspěšně změnit cokoliv jiného kromě svých záznamů v databázi.

| Riziko | Opatření | Pravděpodobnost | Dopad | Kritičnost |
|---|---|---|---|---|
| R2 | Dibi | 0,1 | 80 | 8 |
| R3 | Nette session | 0,25 | 79 | 19,75 |
| R5 | Dibi | 0,1 | 78 | 7,8 |
| R7 | Nette session | 0,25 | 70 | 17,5 |
| R4 | Kontrola na vlastnictví | 0,1 | 60 | 6 |
| R8 | Kontrola na vlastnictví | 0,1 | 62 | 6,2 |

*Tabulka 5 Sumář hodnocení rizik po provedení úprav*

Na následujícím pavučinovém grafu (Obrázek 7.12) je vidět pohyb kritičnosti mnou identifikovaných závažných a kritických rizik. Červenou barvou je označen původní stav a zelenou je označen stav po provedení patřičných úprav a modifikací pro snížení kritičnosti rizik. Jak je vidět, rizika se povedlo do značné míry snížit a proto doufám, že moje aplikace bude do dalších vývojových etap výrazně bezpečnější. Při pohledu to tabulky 5 je možné se povšimnout, že došlo pouze ke snížení pravděpodobnosti výskytu rizika, ale dopad se nezměnil. Důvodem je fakt, že všechna mnou implementovaná opatření mají pouze snížit šanci útočníka na úspěšný útok. Pokud bych chtěl do budoucna snížit i dopad, bylo by nutné přemýšlet o šifrování databáze, případně oddělení databázového a aplikačního serveru, aby v případě útoku nezískal útočník kontrolu nad vším.




*Obrázek 7.12 Pavučinový graf rizik*

## 7.6 PÁTÁ ETAPA

Stejně jako v předchozích etapách i v této jsem začal nejprve opravovat chyby, kterých jsem si všiml při pozorování uživatelů. Na základě zjištění v analýze rizik došlo k přidání kontroly oprávnění pro každou důležitou akci, takže pouhou manipulací URL už nelze modifikovat položky, které uživateli nepatří.

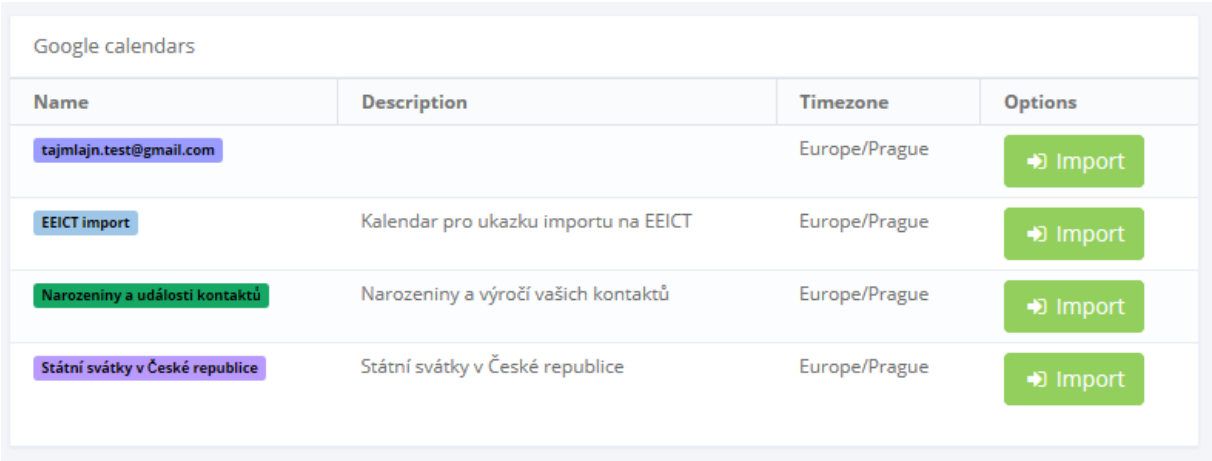
Ke každé události v časové ose bylo přidáno tlačítko pro úpravu, které po kliknutí zobrazí modální dialog a umožní uživateli událost upravit včetně možnosti přesunu do jiné časové osy, kterou vlastní.

Role vedoucího projektu byla nakonec vyřešena velmi jednoduše, kdokoliv založí projekt je zároveň jeho vedoucí a má tak práva na všechny časové osy uvnitř projektu, i když mu nepatří. V souvislosti s tímto oprávněním bylo dána vedoucímu možnost zamykat události. Zamčenou událost nemůže její vlastník nijak upravovat (pokud není zároveň vedoucím projektu). Podnětem pro tuto funkcionalitu byl vedoucí mé diplomové práce, který hodlá používat Tajm-lajn pro plánování práce svých diplomantů a chtěl mít možnost jim naplánovat i události a úkoly, které si nemohou přesunout na později, ale musí je splnit v zadaném termínu. Vedoucí zároveň vidí seznam uživatelů, kteří se projektu účastní a má možnost je z projektu odstranit.

Vzhledem k podpoře pro „veřejné“ časové osy v rámci projektu byl ke každé události přidán malý portrét uživatele, který ji přidal, aby měl vedoucí projektu a všichni jeho účastníci přehled o tom, kdo naplánoval jakou událost a tedy i kdo je za ni zodpovědný.

### 7.6.1 Import událostí z Google kalendáře

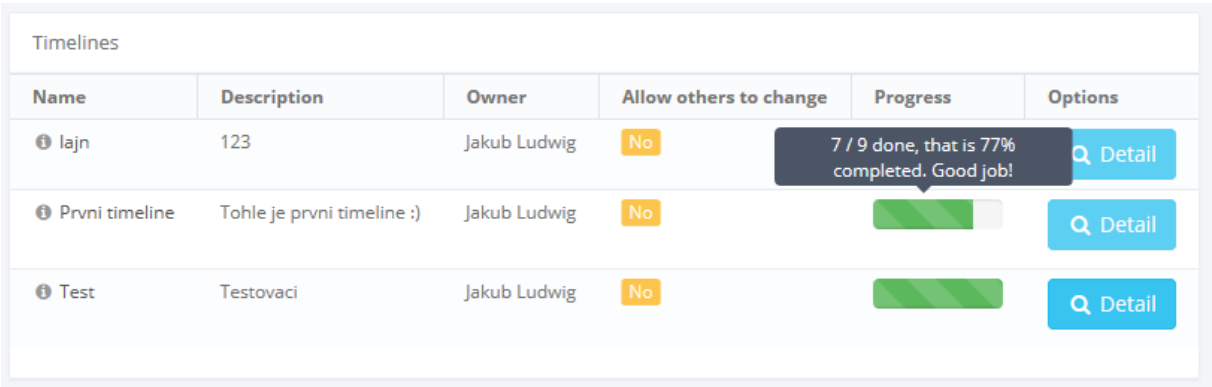
Jelikož vytvářím aplikaci pro plánování času a doufám, že její uživatelé čas tímto ušetří, bylo by zbytečné zatěžovat je ručním přepisováním všech událostí do mé aplikace. Podle vyjádření mých testovacích uživatelů je nejrozšířenější on-line kalendář od firmy Google a vzhledem k tomu, že už samotné přihlašování do mé aplikace vyžaduje Google účet, je dopad nutné administrace na uživatele minimální. Díky jednotnému přihlášení přes OAuth2 a využití kalendářového API společnosti Google jsem vytvořil import událostí, který umožní uživateli výběr konkrétního kalendáře u Google a následně výběr časové osy, do které se mají události nahrát. Uživatel si při importu může zvolit buď existující, nebo novou časovou osu, kterou snadno vytvoří pomocí modálního dialogu, který import nabízí. Barvy kalendářů v nabídce importu korespondují s barvami v Google kalendáři pro lepší přehlednost.




*Obrázek 7.13 Google kalendáře připravené k importu*

### 7.6.2 Zobrazení splněných úkolů

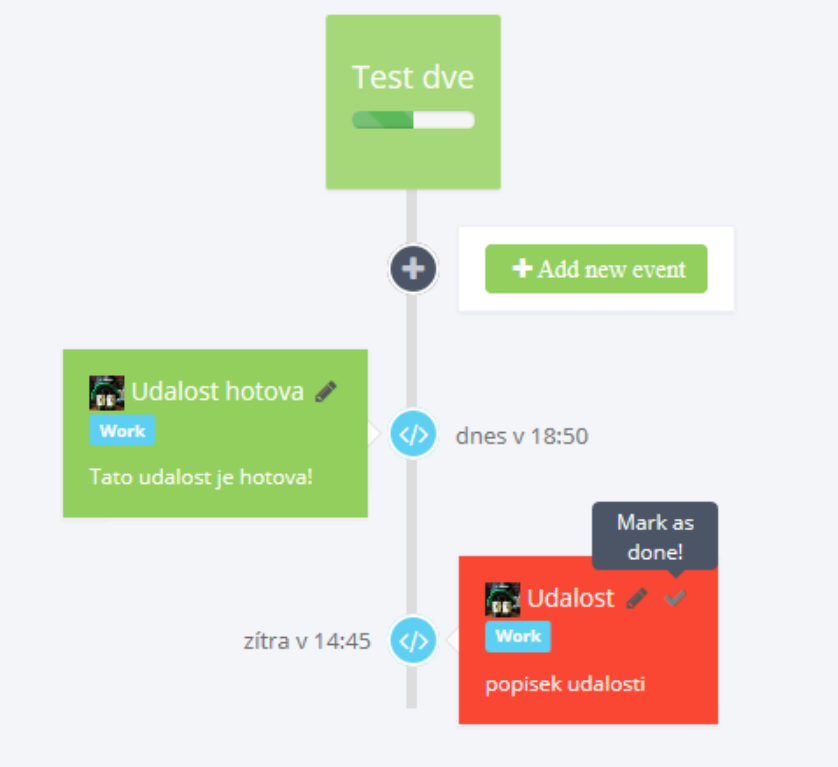
Z pozice vedoucího projektu je aktuálně velmi těžké odhadnout v jakém stavu je práce na projektu, protože mu systém nenabízí žádnou možnost snadného monitoringu průběhu prací. Jeho jediná možnost je ruční projití všech časových os a spočítání, kolik úkolů je hotovo. Z důvodu značné nepohodlnosti tohoto postupu byl do systému přidán ukazatel splnění dané časové osy. Ukazatel se váže na typ události Work, která obsahuje možnost označit ji jako hotovou. Ukazatel je zobrazen v detailu projektu u každé časové osy, jak je zobrazeno na obrázku 8.14.




*Obrázek 7.14 Zobrazení detailu projektu a hotové práce*

Události typu Work jsou v časové ose označeny výraznou červenou nebo zelenou barvou podle svého statusu (Obrázek 7.15). U každé události je navíc tlačítko, kterým se dá

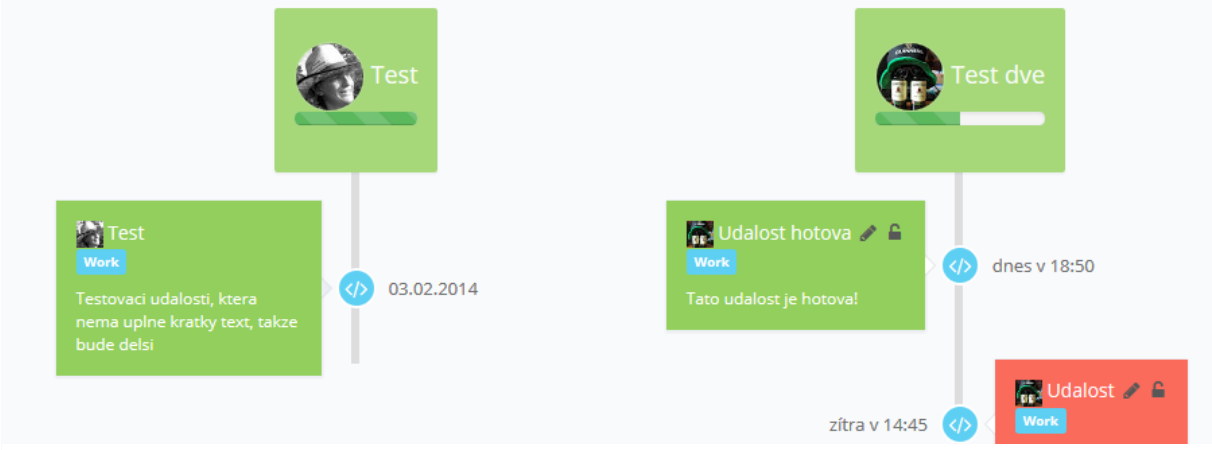
označit jako hotová. Ukazatel v detailu projektu pak funguje jako počitadlo, které zobrazuje počet hotových událostí ke všem událostem typu Work.




*Obrázek 7.15 Ukázka dokončené a nedokončené události*

### 7.6.3 Zobrazení všech časových os projektu

Posledním požadavkem pro tuto etapu bylo zobrazení všech časových os projektu na jedné obrazovce, z důvodu jednoduchého zobrazení a pochopení vazeb mezi jednotlivými




*Obrázek 7.16 Zobrazení všech časových os projektu*

osami členů projektu. Na základě tohoto požadavku jsem vytvořil zobrazení projektu, kde jsou vidět všechny jeho osy. U každé časové osy je z důvodu přehlednosti zobrazen její vlastník pomocí portrétu a jména.

### 7.6.4 Testování

Jelikož se jednalo o poslední vývojovou fázi, kterou budu ve své diplomové práci zpracovávat, tak z výsledků testování vzejde kapitola pojednávající o budoucím vývoji aplikace. Testováním této etapy jsem zjistil, že filtrace v časových osách se ukázala díky zobrazení všech časových os jako nepotřebná a sami uživatelé ji uvedli jako zbytečnou, takže v budoucích verzích bude odstraněna.

Dalším zjištěním je nedokonalost importu událostí z Google kalendáře. Import sice funguje dobře, ale uživatelé často hledali možnost zpětné synchronizace, protože Google kalendář mají například v telefonu a bylo by pro ně pohodlné vidět v kalendáři události naplánované v mé aplikaci.

### 7.6.5 Dotazník

V této etapě došlo ke změně otázek v dotazníku, jelikož se jedná o poslední vývojovou etapu v rámci mé diplomové práce. Dotazníku se zúčastnilo 61 respondentů a odpovídali na následující otázky:

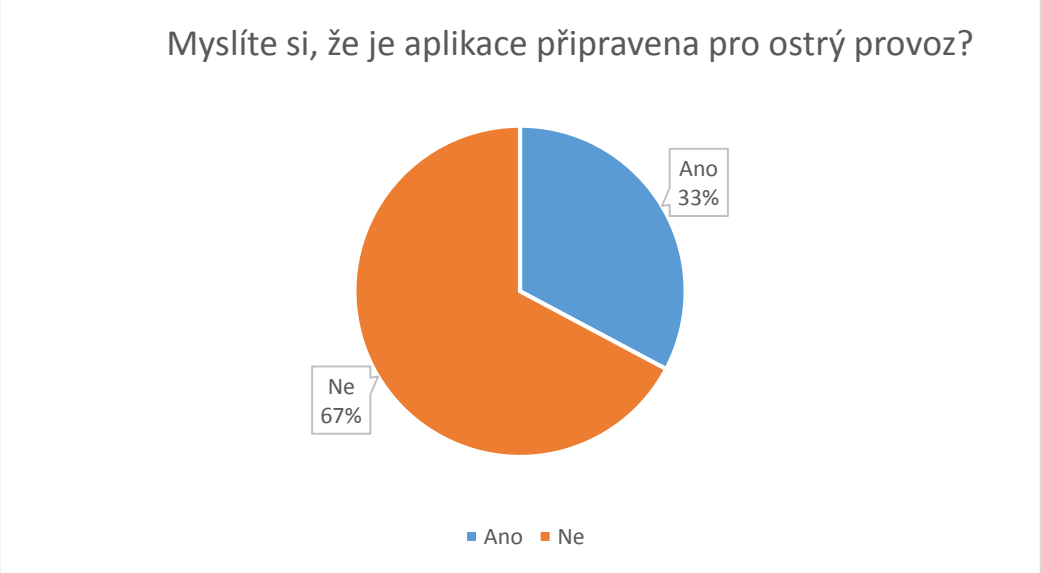
1. Myslíte si, že je aplikace připravena pro ostrý provoz?
2. Jste ochotni za aplikaci v současném stavu zaplatit?
3. Jakou funkčnost mohu do dalších vývojových etap určitě vypustit?
4. Jaká funkčnost naopak rozhodně nesmí chybět v další vývojové etapě?

První dvě otázky měli definované odpovědi na Ano / Ne, druhé dvě otázky měli volnou odpověď, stejně jako tomu bylo v předchozích vývojových etapách. Odpovědi na první dvě otázky zobrazují následující dva grafy.

Odpověď na třetí otázku byla poměrně jednoznačná, jednalo se o filtrování časových os, které se ukázalo s novým zobrazením jako zbytečné. V odpovědích na čtvrtou otázku byli testovací uživatelé překvapivě jednotní a z odpovědí vykrystalizovaly dvě hlavní myšlenky, kterých bych se, podle mé testovací skupiny, měl do budoucna držet. Jednou z nich je možnost synchronizace s Google kalendářem, tedy nepodporovat pouze import, ale synchronizaci oběma směry, kdy při úpravě události v Google kalendáři dojde k úpravě události i v mé aplikaci a naopak. V souvislosti se synchronizací události bude nutné vyřešit kategorizaci událostí,

protože aktuálně je všem importovaným událostem přiřazena kategorie General, což uživatelé nepovažují za nejlepší řešení.

Druhou myšlenkou, která se velmi často opakovala, bylo zavedení mechanismu připomínání událostí uživatelům. Aktuálně aplikace umožňuje plánovat události a různě je členit do časových os, projektů a kategorií, ale chybí mechanismus, který bude uživatelům události aktivně připomínat a nutit je, aby je splnili. Zde je na zvážení, jestli zvolit cestu připomínek, e-mailů a notifikací, nebo spíše cestu motivace pomocí gamifikace.




*Obrázek 7.17 Výsledek první otázky*


*Obrázek 7.18 Výsledek druhé otázky*

# 8 BUDOUCÍ VÝVOJ APLIKACE

V budoucnosti bych rád vyvinul vlastní mobilní aplikaci pro platformu Android, která by poskytovala uživatelům větší komfort, než aktuální responsivní design. Chci se také zaměřit na zapojení gamifikačních principů do celé aplikace a rozdělit ji na dvě části. Jednu s gamifikací, zaměřenou na jednotlivé uživatele a komunity, kde není potřeba tak striktní plánování času, ale jde spíše o organizaci společných zájmů. Tuto verzi bych rád nabízel zdarma. Druhá část mé aplikace už bude komerční a bude určena ať už živnostníkům, nebo malým začínajícím firmám.

Na základě testování v poslední vývojové etapě je jasné, že v nejbližší době potřebuji vyřešit synchronizaci s Google kalendářem, aby se má aplikace dala plnohodnotně používat v kombinaci s tímto velmi rozšířeným nástrojem pro plánování času.

Zapojení gamifikace si do budoucna představuji tak, že vedoucí projektu může naplánovat členům časové osy, kde splnění úkolů přiřadí různé bodové ohodnocení a za splnění celé osy bude další bodové ohodnocení. Uživatelé pak u projektu uvidí žebříčky, jak dobře si vedou a kdo je nejlepší. V ideálním případě by bylo dobré ještě zavést systém odznaků, kdy například za několik úkolů splněných v jednom dnu několik dnů po sobě dostane uživatel speciální odznak, který bude vidět v jeho profilu (například týden v kuse cvičil a chodil běhat). Tyto odznaky by pak definoval vedoucí projektu.

Se systémem odznaků souvisí i zavedení uživatelských profilů, které by měly přehledně zobrazovat uživatelův pokrok v rámci jednotlivých projektů, kterých se účastní. Aktuálně v aplikaci nic takového není, protože jsem se při vývoji soustředil hlavně na samotné časové osy a plánování událostí, ale profily uživatelů jsou velmi důležité, protože díky osobnímu pokroku se dají motivovat ostatní členové projektového teamu.

Do budoucna bude také potřeba modifikovat systém projektů a hlavně správy účastníků projektu. Aplikace aktuálně nabízí pouze jejich výpis a možnost je smazat, ale bude potřeba definovat pro uživatele i různé role a k nim oprávnění (například aby vedoucí mohl delegovat plánování určitého typu událostí).

# ZÁVĚR

Aplikace Tajm-lajn je momentálně ve stavu, kdy splňuje svůj základní cíl, tedy jde pomocí ní plánovat čas. Ovšem je to asi jako tvrdit o poznámkovém bloku, že splňuje svůj základní cíl, protože do něj jde psát. Od základního řešení, které mám teď, bych se v budoucnu rád odrazil a využil zpětnou vazbu od uživatelů k vytvoření produktu, který bude mít šanci na komerční úspěch. A jak potvrzují výsledky posledního dotazníku, mám před sebou ještě hodně práce.

Za dobu věnovanou mé diplomové práci jsem vyvinul aplikaci, která umí nejen plánovat čas, ale obsahuje také mnoho funkcí pro sdílení s ostatními uživateli, je napojená na Google kalendáře, její uživatelské rozhraní je ergonomické a intuitivní a i ve stávající verzi už jsou uživatelé, kteří jí pravidelně využívají. A jejich práce s aplikací není pro účely testování, ale pro opravdové plánování jejich vlastního času. A jejich odezvy jsou velmi pozitivní. Stále je co zlepšovat, ale fakt, že už v této základní verzi je aplikace opravdu používaná, mi dává motivaci v práci pokračovat dál a vyvinout produkt, který mě jednou možná bude živit.

Díky této práci jsem měl možnost poprvé vyzkoušet vyvíjet aplikaci, která od samotného začátku podléhala pravidelnému testování ze strany lidí, kteří ji možná budou v budoucnu používat, a osobně si myslím, že to byl krok tím správným směrem. V minulosti jsem si vyzkoušel opačný způsob, kdy jsem pracoval na aplikaci, kterou zákazník viděl až hotovou, nebo chvíli před dokončením a výsledkem obvykle bylo velké množství předělávání a změn, někdy i v naprostých základech aplikace. To se mi v případě Tajm-lajn nestalo a změny v každé etapě byly minimální. Ano, některé názory testovacích uživatelů se ukázaly jako ne úplně šťastné, například horizontální zobrazení, jehož implementace zabrala několik týdnů a nakonec se ukázalo jako slepá ulička, ale to už patří k tomuto stylu vývoje. Za sebe mohu říct, že do budoucna budu tento způsob vývoje uplatňovat.

# POUŽITÁ LITERATURA


1. LUDWIG, Jakub. *Tajm-lajn* [online]. 2014 [cit. 2014-05-28]. Dostupné z: http://www.tajm-lajn.com
2. THE APACHE SERVER FOUNDATION. *The Apache HTTP Server Project* [online]. 2014 [cit. 2014-05-28]. Dostupné z: http://httpd.apache.org/
3. W3C. *World Wide Web Consortium (W3C)* [online]. 2014 [cit. 2014-05-28]. Dostupné z: http://www.w3.org/
4. THE PHP GROUP. *PHP: Hypertext Preprocessor* [online]. 2014 [cit. 2014-05-28]. Dostupné z: http://php.net/
5. ORACLE CORPORATION. *MySQL :: The world's most popular open source database* [online]. 2014 [cit. 2014-05-28]. Dostupné z: http://mysql.com/
6. The InnoDB Storage Engine. ORACLE CORPORATION. *MySQL :: MySQL 5.5 Reference Manual* [online]. 2014 [cit. 2014-05-28]. Dostupné z: http://dev.mysql.com/doc/refman/5.5/en/innodb-storage-engine.html
7. FACEBOOK. *Facebook* [online]. 2014 [cit. 2014-05-28]. Dostupné z: http://www.facebook.com
8. THE JQUERY FOUNDATION. *JQuery* [online]. 2014 [cit. 2014-05-28]. Dostupné z: http://jquery.com
9. THE JQUERY FOUNDATION. *The jQuery Team* [online]. 2014 [cit. 2014-05-28]. Dostupné z: https://jquery.org/team/
10. ZEND TECHNOLOGIES. *Zend Framework* [online]. 2014 [cit. 2014-05-28]. Dostupné z: http://framework.zend.com/
11. ZEND TECHNOLOGIES. *Zend: The PHP Company* [online]. 2014 [cit. 2014-05-28]. Dostupné z: http://www.zend.com/en/
12. ELLISLAB. *CodeIgniter / EllisLab* [online]. 2014 [cit. 2014-05-28]. Dostupné z: http://ellislab.com/codeigniter
13. ELLISLAB. *EllisLab* [online]. 2014 [cit. 2014-05-28]. Dostupné z: http://ellislab.com/
14. NETTE FOUNDATION. *Rychlý a pohodlný vývoj webových aplikací v PHP | Nette Framework* [online]. 2014 [cit. 2014-05-28]. Dostupné z: http://nette.org/
15. NETTE FOUNDATION. *About the Foundation | Nette Foundation* [online]. 2014 [cit. 2014-05-28]. Dostupné z: http://nettefoundation.com/
16. POTEL, Mike. Potel. *Wildcrest Associates* [online]. 2014 [cit. 2014-05-28]. Dostupné z: http://www.wildcrest.com/Potel/Portfolio/mvp.pdf

17. THE INTERNET ENGINEERING TASK FORCE. *OAuth 2.0* [online]. 2014 [cit. 2014-05-28]. Dostupné z: http://oauth.net/2/
18. OPENID FOUNDATION. *OpenID Foundation website* [online]. 2014 [cit. 2014-05-28]. Dostupné z: http://openid.net/
19. Using OAuth 2.0 to Access Google APIs. GOOGLE. *Google Developers* [online]. 2014 [cit. 2014-05-28]. Dostupné z: https://developers.google.com/accounts/docs/OAuth2
20. MARCOTTE, Ethan a [foreword by Jeremy KEITH]. *Responsive web design*. New York: A Book Apart, 2011. ISBN 978-098-4442-577.
21. TICHÝ, Milík. *Ovládání rizika: analýza a management*. Vyd. 1. Praha: C.H. Beck, 2006, xxvi, 396 s. Beckova edice ekonomie. ISBN 80-717-9415-5.
22. NETTE FOUNDATION. *Dibi* [online]. 2014 [cit. 2014-05-28]. Dostupné z: http://dibiphp.com/en/
23. Sessions. NETTE FOUNDATION. *Nette dokumentace* [online]. 2014 [cit. 2014-05-28]. Dostupné z: http://doc.nette.org/cs/2.1/sessions#toc-bezpecnost-predevsim

# SEZNAM OBRÁZKŮ

# PŘÍLOHA A

Tato příloha obsahuje výsledky jednotlivých dotazníků, které byly předkládány respondentům v rámci jednotlivých vývojových etap. Z důvodů diskrétnosti a vzhledem k faktu, že nemám souhlas se zveřejněním jmen, jsou veškeré výsledky anonymizované.

## PŘEDCHÁZENÍ RIZIKŮM

Respondentům byly položeny tři otázky:

1. Máte celkově pocit, že nestíháte vaše úkoly?
2. Kdyby byl k dispozici nástroj na lepší plánování, máte pocit, že budete úkoly stíhat?
3. Pokud dostanete 10 stejně náročných úkolů, kolik podle vás nestihnete do termínu?

Výsledky reprezentuje následující tabulka, seřazená podle počtu nestihnutých úloh v otázce 3:

| **První otázka** | **Druhá otázka** | **Třetí otázka** |
|---|---|---|
| Ano | Ano | 1 |
| Ano | Ano | 1 |
| Ano | Ano | 1 |
| Ano | Ano | 1 |
| Ano | Ano | 1 |
| Ano | Ano | 1 |
| Ano | Ano | 1 |
| Ano | Ano | 1 |
| Ano | Ano | 1 |
| Ano | Ano | 1 |
| Ano | Ano | 1 |
| Ano | Ano | 1 |
| Ano | Ano | 2 |
| Ano | Ano | 2 |
| Ano | Ano | 2 |
| Ano | Ano | 2 |
| Ano | Ano | 2 |
| Ano | Ano | 2 |
| Ano | Ano | 2 |
| Ano | Ano | 2 |
| Ano | Ano | 2 |
| Ano | Ano | 2 |

| | | |
|---|---|---|
| Ano | Ano | 3 |
| Ano | Ano | 3 |
| Ano | Ne | 3 |
| Ano | Ne | 3 |
| Ano | Ne | 3 |
| Ano | Ne | 3 |
| Ano | Ne | 3 |
| Ano | Ne | 3 |
| Ano | Ne | 3 |
| Ano | Ne | 3 |
| Ano | Ne | 4 |
| Ano | Ne | 4 |
| Ano | Ne | 4 |
| Ano | Ne | 4 |
| Ano | Ne | 4 |
| Ano | Ne | 4 |
| Ano | Ne | 4 |
| Ano | Ne | 4 |
| Ano | Ne | 6 |
| Ano | Ne | 6 |
| Ano | Ne | 7 |
| Ano | Ne | 7 |
| Ne | Ne | 7 |
| Ne | Ne | 7 |
| Ne | Ne | 8 |
| Ne | Ne | 8 |
| Ne | Ne | 10 |
| Ne | Ne | 10 |

## DOTAZNÍK DRUHÉ VÝVOJOVÉ ETAPY

Respondenti byli dotazováni na to, co by se jim v aplikaci dále líbilo.

| **Možnost** | **Počet uživatelů** |
|---|---|
| Zobrazení času lepším způsobem | 10 |
| Zahrnout délku trvání události | 12 |
| Zobrazit i události z minulosti | 8 |
| Přidat podporu pro projekty | 15 |
| Označit úkol jako splněný | 16 |

## DOTAZNÍK TŘETÍ VÝVOJOVÉ ETAPY

Respondenti byli dotazováni na to, co by se jim v aplikaci dále líbilo.

| Možnost | Počet uživatelů |
|---|---|
| Sdílení projektů mezi uživateli | 25 |
| Filtrování | 12 |
| Více typů událostí | 26 |
| Povolení upravovat osu | 13 |
| Více uživatelů v jednom projektu | 20 |

## DOTAZNÍK ČTVRTÉ VÝVOJOVÉ ETAPY

Respondenti byli dotazováni na to, co by se jim v aplikaci dále líbilo.

| Možnost | Počet uživatelů |
|---|---|
| Zobrazit počet splněných úkolů | 30 |
| Zobrazit všechny osy v projektu | 28 |
| Import z Google kalendáře | 35 |
| Editace událostí | 41 |
| Role vedoucího projektu | 31 |

## RIZIKOVÁ ANALÝZA

Pro rizikovou analýzu byli dotázání tři experti a každý z nich dostal formulář, kde se měl vyjádřit k mnou identifikovaným rizikům v kapitole 7.5. Každá z tabulek reprezentuje jednoho ze třech dotázaných expertů.

| Riziko | Pravděpodobnost | Dopad |
|---|---|---|
| R1 | 0.08 | 75.00 |
| R2 | 0.35 | 72.00 |
| R3 | 0.23 | 82.00 |
| R4 | 0.72 | 63.00 |
| R5 | 0.62 | 81.00 |
| R6 | 0.42 | 45.00 |
| R7 | 0.36 | 65.00 |
| R8 | 0.74 | 60.00 |

| Riziko | Pravděpodobnost | Dopad |
|---|---|---|
| R1 | 0.13 | 85.00 |
| R2 | 0.41 | 81.00 |
| R3 | 0.41 | 71.00 |
| R4 | 0.85 | 68.00 |
| R5 | 0.61 | 82.00 |
| R6 | 0.32 | 45.00 |
| R7 | 0.31 | 66.00 |
| R8 | 0.85 | 65.00 |

| Riziko | Pravděpodobnost | Dopad |
|---|---|---|
| R1 | 0.09 | 86.00 |
| R2 | 0.74 | 87.00 |
| R3 | 0.41 | 84.00 |
| R4 | 0.83 | 49.00 |
| R5 | 0.27 | 71.00 |
| R6 | 0.46 | 36.00 |
| R7 | 0.23 | 79.00 |
| R8 | 0.81 | 61.00 |

## DOTAZNÍK PÁTÉ VÝVOJOVÉ ETAPY

Respondenti byli dotazováni na to, co by se jim v aplikaci dále líbilo a zda považují aplikaci za připravenou pro komerční provoz.

| Myslíte si, že je aplikace připravena pro ostrý provoz? | |
|---|---|
| Ano | 20 |
| Ne | 41 |

| Jste ochotni za aplikaci v současném stavu zaplatit? | |
|---|---|
| Ano | 15 |
| Ne | 46 |